INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Segurança de Redes em Sistemas de Experimentação Remota

VICTOR JOÃO SEIXAS ALVES
Setembro de 2012

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Segurança de Redes em Sistemas de Experimentação Remota

Victor João Seixas Alves

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Arquitectura, Sistemas e Redes

Orientador: Professor Doutor Manuel Gradim de Oliveira Gericota

Co-orientador: Mestre Paulo Alexandre Duarte Ferreira

Júri:

Presidente:

[Nome do Presidente, Escola]

Vogais:

[Nome do Vogal1, Escola]

[Nome do Vogal2, Escola] (até 4 vogais)

Porto, Setembro 2012

Resumo

O crescimento dos sistemas de informação e a sua utilização massiva criou uma nova realidade no acesso a experiências remotas que se encontram geograficamente distribuídas. Nestes últimos tempos, a temática dos laboratórios remotos apareceu nos mais diversos campos como o do ensino ou o de sistemas industriais de controlo e monitorização.

Como o acesso aos laboratórios é efectuado através de um meio permissivo como é o caso da Internet, a informação pode estar à mercê de qualquer atacante. Assim, é necessário garantir a segurança do acesso, de forma a criar condições para que não se verifique a adulteração dos valores obtidos, bem como a existência de acessos não permitidos. Os mecanismos de segurança adoptados devem ter em consideração a necessidade de autenticação e autorização, sendo estes pontos críticos no que respeita à segurança, pois estes laboratórios podem estar a controlar equipamentos sensíveis e dispendiosos, podendo até eventualmente comprometer em certos casos o controlo e a monotorização de sistemas industriais.

Este trabalho teve como objectivo a análise da segurança em redes, tendo sido realizado um estudo sobre os vários conceitos e mecanismos de segurança necessários para garantir a segurança nas comunicações entre laboratórios remotos. Dele resultam as três soluções apresentadas de comunicação segura para laboratórios remotos distribuídos geograficamente, recorrendo às tecnologias IPSec, OpenVPN e PPTP.

De forma a minimizar custos, toda a implementação foi assente em software de código aberto e na utilização de um computador de baixo custo. No que respeita à criação das VPNs, estas foram configuradas de modo a permitir obter os resultados pretendidos na criação de uma ligação segura para laboratórios remotos. O pfSense mostrou-se a escolha acertada visto que suporta nativamente quaisquer das tecnologias que foram estudadas e implementadas, sem necessidade de usar recursos físicos muito caros, permitindo o uso de tecnologias de código aberto sem comprometer a segurança no funcionamento das soluções que suportam a segurança nas comunicações dos laboratórios remotos.

Palavras-chave: Laboratórios remotos, segurança, IPSec, OpenVPN, PPTP, VPN, pfSense

Abstract

The increased use of information systems enabled the easy access to geographically distributed laboratories. Recently, the use of remote laboratories is being mentioned in such distinct areas as training institutions or industrial systems control and monitoring.

Since access to remote laboratories is done using a permissive environment such as the Internet, information exchange can be at the mercy of any attacker. Thus, it is necessary to ensure that security mechanisms are in place to avoid data corruption or unauthorized accesses to the remote laboratories. They should take into account the need for authentication and authorization, which are critical with respect to safety, because these laboratories may enable the access and control of expensive equipment, and the monitoring and control of industrial systems, whose security may not be compromised.

The aim of this study is to analyze various network security mechanisms able to ensure safe and reliable data transfer among remote laboratories as well as secure access control. Three solutions making use of IPSec, PPTP and OpenVpn are discussed. To minimize costs, the entire implementation was based on open source software and the use of a low-cost computer. In the case of VPNs, they are configured to obtain a secure connection to remote laboratories.

The conclusion is that pfSense is the right choice since it natively supports the totality of the technologies studied and implemented without the need of very expensive physical resources, and enabling the use of open source technologies without compromising the secure access to remote laboratories.

Keywords: Remote Laboratories, security, IPSec, OpenVpn, PPTP, VPN, pfSense

Agradecimentos

A escrita de uma dissertação é um trabalho de árdua dedicação.

Gostaria de agradecer ao Professor Manuel Gericota pela oportunidade de desenvolvimento deste trabalho.

Ao Professor Paulo Ferreira pelo apoio e ajuda prestados a favor do bom desenvolvimento deste projecto

Ao Laboris a disponibilidade do material e espaço necessário para a realização do mesmo.

Aos meus colegas e Professores do ISEP que duma forma directa e indirecta contribuíram para a realização deste trabalho.

Agradeço aos meus pais toda a dedicação e carinho e são um grande incentivo para todo o meu esforço.

À minha filha Margarida pelo seu belo sorriso que ajuda a superar tudo.

Agradeço a uma pessoa muito especial, a minha esposa Simone, pela ajuda e incentivo mesmo nos momentos mais difíceis.

Obrigado a todos.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

3DES	*Triple Data Encryption Algorithm*
ACK	*Acknowledgement Packet*
AES	*Advanced Encryption Standard*
AH	*Authentication Header*
ARP	*Address Resolution Protocol*
ARPANET	*Advanced Research Projects Agency Network*
AS	*Autonomous System*
ASCII	*American Standard Code for Information Interchange*
ATM	*Asynchronous Transfer Mode*
BF-CBC	*Blowfish-Cipher Block Chaining*
BITS	*Bump In The Stack*
BITW	*Bump In The Wire*
CA	*Certificate Authority*
CA	*Certification Authority*
CCP	*Compression Control Protocol*
CPS	*Certification Practice Statements*
CRC	*Cyclic Redundancy Check*
CRL	*Certificate Revocation Lists*
CSMA/CD	*Carrier Sense Multiple Access with Collision Detection*
DARPA	*Defense Advanced Research Project Agency*
DES	*Data Encryption Standard*
DHCP	*Dynamic Host Configuration Protocol*
DI	*Domain Interpretation*

DNS *Domain Name System*

DoS *Denial of Service*

DSA *Digital Signature Algorithm*

ECI *ECI Telematics*

ECP *Encryption Control Protocol*

ESP *Encapsulating Security Payload*

FIN *Finish (flag)*

FTP *File Transfer Protocol*

GRE *Generic Routing Encapsulation*

HMAC *Hash based Message Authentication Code*

HMACSHA1 *Hash based Message Authentication Code using Secure Hash Algorithm*

HTTP *Hyper Text Transfer Protocol*

HTTPS *Hyper Text Transfer Protocol Secure*

IAB *Internet Activity Board*

IANA *Internet Assigned Numbers Authority*

IBM *International Business Machines*

ICMP *Internet Control Message Protocol*

IEEE *Institute of Electrical and Electronics Engineers*

IETF *Internet Engineering Task Force*

IGMP *Internet Group Management Protocol*

IKE *Internet Key Exchange*

IP *Internet Protocol*

IPSec *IP Security Protocol*

IPv4 *Internet Protocol version 4*

IPv6 *Internet Protocol version 6*

ISAKMP *Internet Security Association Management*

ISP	*Internet Service Provider*
IT	*Information Technology*
ITU	*International Telecommunications Union*
L2F	*Layer 2 Forwarding*
L2TP	*Layer 2 Protocol Tunneling*
LAN	*Local Area Network*
LDAP	*Lightweight Directory Access Protocol*
LLC	*Logical Link Control*
LZO	*Lempel-Ziv-Oberhume Compression*
MAC	*Media Access Control address*
MAN	*Metropolitan Area Network*
MD	*Message Digest Algorithm*
MD2	*Message Digest Algorithm 2*
MD4	*Message Digest Algorithm 4*
MD5	*Message Digest Algorithm 5*
MTU	*Maximum Transport Unit*
NAT	*Network Address Translation*
NIC	*Network Interface Card*
OCSP	*Online Certificate Status Protocol*
OSI	*Open Systems Interconnection*
PGP	*Pretty Good Privacy*
PIN	*Personal Identification Number*
PKI	*Public Key Infrastructure*
PPP	*Point to Point Protocol*
PPTP	*Point to Point Tunneling Protocol*
RA	*Registration Authority*

RC4	*Ron's Code 4*
RFC	*Request for Comments*
RIPEMD	*Race Integrity Primitives Evaluation Message Digest*
RSA	*Rivest, Shamir, & Adleman*
SA	*Security Association*
SACK	*Selective ACK*
SAN	*Storage Area Network*
SET	*Secure Electronic Transaction*
SHA	*Secure Hash Algorithm*
SKIP	*Simple Key Management for IP*
SMTP	*Simple Mail Transfer Protocol*
SPI	*Security Parameter Index*
SSH	*Secure Shell*
SSL	*Secure Sockets Layer*
SYN	*Synchronize*
SYN/ACK	*Synchronize Acknowledge*
TCP	*Transmission Control Protocol*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
TELNET	*Network Virtual Terminal Protocol*
TLS	*Transport Layer Security*
TTL	*Time to Live*
TTP	*Trusted Third Party*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*
VoIP	*Voice over Internet Protocol*
VPN	*Virtual Private Network*

WAN *Wide Area Network*

WEP *Wired Equivalent Privacy*

1 Introdução

A temática dos laboratórios remotos começa a ser uma realidade, motivada pela massificação da sua utilização. Esta utilização massiva coloca problemas relacionados com a segurança nas redes para acesso a laboratórios remotos.

A segurança das redes informáticas é hoje um ponto de extrema relevância no que respeita à transmissão de dados entre experiências dentro da rede local e entre redes com localizações geográficas diferentes.

Os mecanismos de segurança adoptados devem ter em consideração a necessidade de autenticação e autorização. No caso do primeiro mecanismo, este é necessário para que exista a garantia de identificação do emissor e do receptor para que a comunicação seja sempre estabelecida entre as partes interessadas. No caso do segundo, é o garante da protecção contra acções não autorizadas. Independentemente da forma pela qual os dados são acedidos e do meio através do qual são partilhados ou armazenados, recomenda-se que sejam sempre protegidos.

A segurança dos dados na rede é obtida a partir da implementação de uma série de controlos que podem ser de origem política, prática, procedimental ou de estrutura organizacional.

No caso dos laboratórios remotos, no acesso a uma experiencia remota é necessário garantir a segurança do acesso, de forma a criar condições para que não se verifique a adulteração dos valores obtidos, bem como acessos que não sejam permitidos. É necessário também garantir que quem executa as acções é verdadeiramente quem se afirma ser, e por fim garantir um registo sobre as acções executadas nos laboratórios remotos.

Esta dissertação faz uma abordagem a vários conceitos e mecanismos de segurança necessários para garantir a segurança das redes que acedem a laboratórios remotos, culminando na proposta e apresentação de três soluções que garantem essa segurança.

Os meios tecnológicos usados neste estudo incluem uma *firewall* em código aberto que disponibiliza nativamente quatro tipos distintos de *VPN (Virtual Private Network)*. No decorrer

da implementação foram estudadas três dessas tecnologias, cuja utilização tem como objectivo garantir a segurança nas redes que permitem o acesso a experiências em laboratórios remotos, utilizando meios permissivos como é o caso da Internet.

1.1 Objectivos

Actualmente, no ensino em geral e no ensino nas áreas da engenharia em particular, é cada vez mais necessária a utilização de laboratórios remotos, os quais possibilitam o acesso através da Internet a um leque alargado de experiências localizadas em locais distintos, sem restrições de horários.

Os principais objectivos desta dissertação são:

- Estudo e análise das ameaças gerais à segurança em redes;
- Estudo de mecanismos de segurança;
- Estudo de algoritmos de criptografia;
- Estudo de protocolos de rede seguros;
- Estudo de tecnologias de redes privadas virtuais;
- Criação de uma solução baseada em *software* de código aberto para o suporte de laboratórios remotos distribuídos, permitindo a interligação de experiências de uma forma segura, usando a Internet;
- Implementação de três protótipos usando tecnologias distintas de redes privadas virtuais.

1.2 Estrutura

Estruturalmente esta dissertação está dividida em nove capítulos, sendo que neste primeiro capítulo é realizada uma introdução à dissertação e definidos os objectivos e estrutura da dissertação.

No segundo capítulo são realizadas introduções ao tema laboratórios remotos. São igualmente abordadas as vantagens e desvantagens dos laboratórios remotos, proporcionada uma explicação resumida sobre o desenvolvimento de um laboratório remoto.

No terceiro capítulo são introduzidos conceitos do *TCP/IP (Transmission Control Protocol/Internet Protocol)*, alguns conceitos relacionados com as redes de dados, conceitos

genéricos sobre redes e, por fim, é abordado o conjunto de protocolos *TCP/IP*, sendo analisados a sua estrutura e as camadas e os protocolos que o compõem.

No quarto capítulo é abordado a questão da segurança informática, abordando-se temas como as ameaças à segurança e os mecanismos de segurança. No quinto capítulo são referidas algumas noções de criptografia e abordados os algoritmos de cifra.

No sexto capítulo é realizado um estudo sobre protocolos de redes seguros, com ênfase no *IPSec (IP Security Protocol)*. No sétimo capítulo são abordados conceitos genéricos de redes privadas virtuais e protocolos.

No oitavo capítulo são descritos os requisitos da implementação, arquitectura da solução, escolha da solução e implementação das três soluções.

Finalmente, no nono capítulo, são apresentadas algumas conclusões sobre o trabalho desenvolvido, tais como a análise contrastiva entre os resultados e os objectivos iniciais propostos para o trabalho, e perspectivadas possíveis linhas a seguir para trabalho futuro, rematando-se com algumas considerações finais.

2 Introdução aos Laboratórios Remotos

A inclusão das Tecnologias de Informação e Comunicação (*TIC*) no sistema de ensino e aprendizagem difunde novas formas de aprender, de leccionar e de raciocinar. Nos derradeiros anos, temos presenciado várias tentativas para incluir recursos educativos na *Internet*, de forma a beneficiar de muitas das capacidades que estes novos ambientes oferecem. (Junior, 2007)

Uma destas inclusões são os laboratórios remotos, que são disponibilizados por meio da *Internet* e que possibilitam o desenvolvimento remoto de experiências laboratoriais. Neste segundo capítulo aborda-se a temática dos laboratórios remotos. O estudo dos laboratórios remotos nas áreas de engenharia iniciou-se com os laboratórios de controlo e automação de experiências. Dada a necessidade dos utilizadores terem acesso remoto aos equipamentos, as experiências foram adaptadas para possibilitarem a utilização de *robots* na manipulação de aparelhos. (Junior, 2007)

A utilização de ambientes remotos tem como objectivos, trazer comodidade, segurança, economia e um maior aproveitamento dos recursos livres. O acesso remoto permite o controlo de diversas tarefas, como por exemplo, o controlo de electrodomésticos e sensores, accionamento de sistemas de alarmes e sistemas de segurança. (Junior, 2007)

Um laboratório remoto permite a interacção com processos reais (Santos et al., 2004) possibilitando assim ao utilizador um contacto com dilemas práticos do mundo real, ausentes dos laboratórios virtuais ou de ambientes de simulação.

2.1 Vantagens e desvantagens dos laboratórios remotos

Os laboratórios remotos são aqueles em que os instrumentos e as experiências são reais, sendo apenas o acesso virtual. Assim sendo, usando laboratórios remotos obtêm-se as seguintes vantagens: (Nedic et al., 2003)

- Existe interacção directa com equipamentos reais;
- A informação é real;
- Não existem restrições quer ao nível do tempo nem do espaço;
- Há a possibilidade do acompanhamento das experiências em tempo real, ou da sua análise em tempo diferido, usando uma ligação Internet.

Porém, existem as seguintes desvantagens: (Nedic et al., 2003)

- A presença apenas virtual no laboratório;
- Problemas colaborativos entre os colegas e os professores;
- Problema no acesso simultâneo à mesma experiência.
- Carece de alta tecnologia;
- Requer a compra de *software* de desenvolvimento dispendioso;
- Necessita de uma infra-estrutura com muitos elementos de *hardware* que possibilitem o desenvolvimento do mecanismo de captura e transferência das imagens do laboratório remoto;
- Maior largura de banda, para transmissão dos dados do laboratório.

2.2 Desenvolvendo um laboratório remoto

O desenvolvimento de um laboratório remoto é um processo complexo, pois mistura questões técnicas de *hardware* e *software*. O primeiro passo é a concepção de uma experiência num laboratório real, à qual está agregada um conjunto de instrumentos controláveis por computador, o que, no caso mais simples, pode ser apenas uma placa acessível e controlável remotamente. (Junior, 2007)

Esta estrutura facilitará a alimentação do equipamento e o accionamento das funções de início e fim de envio dos resultados para a janela onde a experiência está a ser mostrada ao utilizador.

Algumas etapas deverão ser seguidas no desenvolvimento do laboratório remoto. São elas: (Teixeira et al., 2005)

- Elaboração do material teórico de forma a permitir um processo de estudo mais eficiente e célere;
- Explicar os temas de aprendizagem;
- Fundamentar cada figura ou diagrama gráfico possibilitando o estudo individual;
- Definir um agrupado de questões permitindo a apreciação dos conhecimentos obtidos;

- Ilustrar a aparência gráfica, possibilitando ao aluno uma boa aproximação à realidade;
- Escolha das linguagens de programação, por exemplo o *Labview* e *Java*;
- Oferecer a ligação remota ao laboratório real, abrangendo o *hardware* necessário, como por exemplo placas, câmara *Web* e cabos;
- Disponibilização de ficheiros em servidor *Web*;
- Teste das experiências remotas a fim de encontrar possíveis falhas e inconsistências no acesso remoto ao laboratório.

Na Figura 2.1, pode-se observar um exemplo de um acesso a um laboratório remoto.

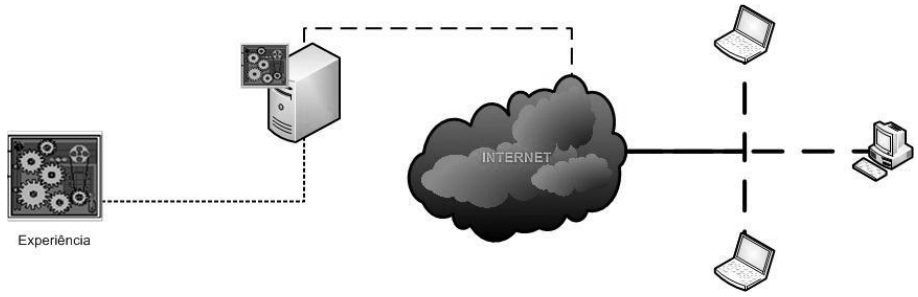

Figura 2.1 - Esquema de acesso a um laboratório remoto

O acesso é realizado através da *Internet* e o utilizador acede ao laboratório remoto que se encontra disponível num servidor. A partir desse momento ocorre o accionamento do sistema que receberá os dados do utilizador. Neste momento também ocorre o posicionamento da câmara e a inicialização de outros instrumentos necessários ao funcionamento do laboratório. Após a calibração das câmaras, a câmara transmite ao utilizador tudo o que se passa na experiência, assim como vão sendo adquiridos e guardados ou enviados os dados do sistema em execução. (Junior, 2007)

2.3 Conclusão

De forma a concluir o ponto referente aos laboratórios remotos, foi abordada uma introdução aos laboratórios remotos, vantagens e desvantagens e por fim, descrito o processo de desenvolvimento de um laboratório remoto. No próximo capítulo são abordados o protocolo *TCP/IP* e alguns conceitos de redes.

3 Introdução ao TCP/IP e Conceitos de Redes

3.1 Noções básicas

Neste ponto são abordadas algumas questões relacionadas com o protocolo *TCP/IP*. Existem vários conceitos de rede, a qual pode ir de uma simples ligação entre dois computadores, até ao mundo complexo da *Internet*. Antes de avançar no tema será útil definir alguns conceitos. (Loshin, 2004)

- **Entidade** - é qualquer objecto que faz algo. Dependendo do contexto, uma entidade pode ser uma pessoa, um computador, um programa, um dispositivo ou um processo.

- **Sistema** - É uma qualquer entidade com um comportamento observável e reproduzido, ou por vezes também chamado de caixa negra. Um sistema pode ser um componente de *hardware* ou um processo a correr num componente de *hardware*. O sistema aceita entradas e produz saídas. Dentro de um sistema podem existir outros sistemas.

- **Rede** - Um conjunto de sistemas que partilham o mesmo meio de comunicação. Mas pode não ser suficiente partilhar o meio. Terão de ter regras que permitam a comunicação (protocolos). Uma rede também é um sistema.

- **Protocolo** - Um conjunto de regras que definem como um sistema interage.

- ***Internet*** - Um conjunto de redes, que faz dela uma rede também. Pode utilizar um meio físico único, ou pode abranger diferentes meios físicos, usando para isso um meio virtual.

- **Meio** - É o meio físico por onde os sinais são transferidos. Pode ser um cabo de rede ou um meio sem fios. Depende sempre de uma infra-estrutura. Por exemplo, numa rede sem fios será necessário um ponto de acesso, um comutador, repetidores de sinal, entre outros.

- **Interface** - O ponto onde duas entidades realizam o contacto.

- **Nó** - Uma qualquer entidade que está ligada a uma rede e é capaz de criar e usar dados da rede.

- ***Host*** - Qualquer nó que suporte utilizadores e corra aplicações. Um *host* e um nó por vezes são usados como sinónimos, mas são dois tipos de entidades diferentes.

- **Cliente** - Uma entidade de rede que requer serviços de rede de um servidor.

- **Servidor** - Uma entidade de rede que responde a pedidos dos clientes.

- ***LAN (Local Area Network)*** - É uma rede local, utilizada para distâncias pequenas, abrangendo por norma salas, pisos ou edifícios.

- ***WAN (Wide Area Network)*** - Uma rede que é usada para ligar longas distâncias. Por exemplo, a ligação de uma sede a uma filial separada por centenas de quilómetros.

Para uma rede existir terão de existir três objectos: um recurso de rede; entidades ligadas ao longo do meio; e interfaces através das quais as entidades passam as mensagens.

3.1.1 Meio

O meio da rede é a parte física ou o mecanismo através do qual os sinais de rede são enviados. Podemos ter redes com fios ou redes sem fios. As primeiras baseiam-se em cabos de cobre ou fibra-óptica, pelos quais são transportados os sinais. No caso das redes sem fios, estes são transportados por ondas de rádio. (Loshin, 2004)

Apesar de utilizarem meios diferentes todas as redes têm de ter um conjunto de protocolos que definem o seu funcionamento (regras). Um qualquer sistema ligado numa rede *Ethernet* deverá ser capaz de distinguir o ruído de uma transmissão de dados. O termo ligação é muitas vezes usado para se referir a um meio e ao conjunto de regras que regem as transmissões nesse meio. (Loshin, 2004)

3.1.2 Interfaces

Pode-se considerar que uma placa de rede fornece o contacto entre um computador e uma rede *LAN*. Assim, o nó transfere os dados para a placa de rede, que os transforma em sinais electrónicos e envia-os para o cabo. (Loshin, 2004)

3.1.3 Nó e *Host*

Qualquer objecto ligado a uma rede e que seja capaz de criar dados na rede e usá-los é apelidado de nó. Por exemplo, uma impressora com uma interface de rede é considerada um nó, caso aceite e processe pedidos que sejam transmitidos através da rede, e consiga produzir

os seus próprios dados, que retornam códigos de conclusão dos trabalhos para o serviço de impressão. (Loshin, 2004)

Um computador devidamente ligado a uma rede, e onde é executado *software* de rede, é considerado um nó, pois pode transmitir e receber dados através de uma rede. Um comutador não pode ser considerado um nó, pois não cria os seus próprios dados, apenas transmitindo dados de outros. (Loshin, 2004)

Existe a noção que *host* e nó são sinónimos, mas esta ligação não é estritamente correcta. Um nó é um objecto que emite e absorve dados da rede (impressora de rede), enquanto um *host* é um qualquer nó que suporta um ou mais utilizadores e que corre um *software* de rede. (Loshin, 2004)

Por exemplo, uma impressora pode ser considerada um *host* se permitir autenticação de utilizadores e tiver um aplicativo que permita, por exemplo, ver a fila de impressão e o seu estado. (Loshin, 2004)

3.1.4 Clientes e Servidores

O termo cliente-servidor remonta aos tempos que a rede significava a ligação de clientes (computador, terminais) a servidores *mainframes*. O cliente usa mecanismos específicos para aceder ao servidor, e o servidor garante esses pedidos. Nos dias de hoje o cenário alterou-se: servidor pode enviar, bem como receber dados dos clientes. Por vezes pode ser difícil saber qual o cliente e qual o servidor. Por exemplo, o serviço de *DNS (Domain Name System)* é disso um bom exemplo: o mesmo nó pode agir como cliente e como servidor. (Loshin, 2004)

3.1.5 Tipos de redes

Com o desenvolvimento das redes de dados, vários termos têm surgido para diferenciar os vários tipos de redes que existem. Normalmente estes termos estão associados às dimensões das redes, sendo os termos *LAN*, *WAN*, *MAN (Metropolitan Area Network)* e *SAN (Storage Area Network)*. (Loshin, 2004)

Quando falamos no caso das redes *LAN*, estamos a falar em pequenas distâncias, como, por exemplo, uma sala, um piso ou um edifício, normalmente redes restritas a um local. As ligações são normalmente realizadas por cabo ou sem fios (*wireless*) e limitadas no número de utilizadores ligados. Este tipo de redes é normalmente gerido pela organização que as utiliza, ao contrário das *WAN* que são geridas por terceiros, normalmente operadores de telecomunicações. (Loshin, 2004)

Vejamos o caso de uma organização internacional com uma sede e vários escritórios espalhados pelo globo. A gestão da rede local cabe às equipas de *IT*. No que diz respeito às

ligações entre as suas filiais e a sede, é adquirido um serviço de comunicações que garante a interconectividade entre os vários sítios. (Loshin, 2004)

As redes *MAN*, são uma abordagem que fica no meio das redes *LAN* e *WAN*, sendo maiores que as *LAN* e mais pequenas que as *WAN*. A título de exemplo, uma *MAN* será a rede que abrange um *campus* universitário, ou seja, as ligações entre edifícios distanciados por alguns quilómetros, podendo ser realizada através de ligações de micro-ondas. (Loshin, 2004)

As ligações usadas neste tipo de redes podem ser semelhante às usadas nas *WAN*, contratando circuitos a um operador de telecomunicações, ou mesmo, usar tecnologias de *LAN* para a ligação de alguns pontos. A nomenclatura *SAN*, foi recentemente adicionada às siglas de rede, referindo-se às ligações de rede entre dispositivos de armazenamento de dados. (Loshin, 2004)

3.1.6 Sistemas de rede

Para o conceito de rede é importante compreender como as redes se interligam, e compreender também o comportamento de redes individuais que se podem comportar como sistemas e sistemas que se podem comportar como redes.

Para compreender o conceito olhemos para uma caixa negra, a qual aceita entradas de várias origens, e tem várias maneiras de produzir saídas. O que ocorre dentro da caixa é indiferente para o resultado dos dados obtidos. (Loshin, 2004)

Olhemos para o exemplo de um sistema autónomo (*AS (Autonomous System)*). Trata-se de uma rede que tem um comportamento delimitado. O tráfego que entra num *AS* pode ser considerado como tráfego a entrar numa caixa negra, o que ocorre dentro do *AS* não é relevante para ninguém fora dele.

A Figura 3.1 mostra as várias maneiras de mapear uma rede num *AS*. A rede **A** é um *AS*, bem como as redes **B** e **C**. Embora a figura apresente três redes distintas de sistemas autónomos, se se olhar a partir da rede **C** dá a ideia que existem apenas duas, que são as redes **B** e **C**, pois a rede **A** parece pertencer à rede **B**. Mas se se olhar a partir da rede **B**, fica claro que existem três redes distintas, funcionado a rede **B** como uma rede de interligação para redes distintas. (Loshin, 2004)

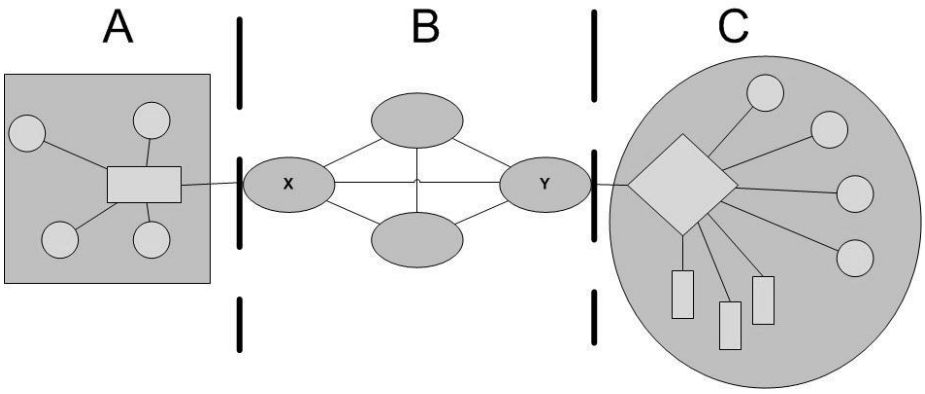

Figura 3.1 - Sistemas Autónomos (Loshin, 2004)

3.2 Protocolos de rede

Numa comunicação entre dois sistemas, tem de existir um conjunto de regras, as quais vão possibilitar a troca de informação entre pontos distintos na rede. Numa rede alargada como a *Internet*, existem sistemas distintos que comunicam entre si, sendo que, por isso, existe a necessidade de obter interoperabilidade entre eles, pelo que têm de existir regras que possibilitem tal. A interoperabilidade é a capacidade que os sistemas têm para comunicar e trabalhar em conjunto, mesmo não tendo qualquer informação acerca uns dos outros. Para que isso aconteça apenas utilizam um conjunto de normas. O *TCP/IP* tem como um dos principais objectivos possibilitar essa mesma interoperabilidade, sem necessidade de distinção de qualquer tipo de *hardware* e *software*. (Loshin, 2004)

No cenário da rede *Internet*, o ideal é tornar possível a todos os utilizadores a partilha de informação entre eles, sem que estes tenham de estar preocupados com o que se passa no outro extremo da comunicação, como por exemplo, qual ou quais os sistemas operativos, os tipos de *hardware* utilizados ou o *software*. Um protocolo, como já foi dito, é um conjunto de regras que definem uma interacção entre dois ou mais sistemas, possibilitando assim a comunicação entre eles. (Loshin, 2004)

Segundo o autor (Loshin, 2004), um protocolo deve especificar:

Como as entidades iniciam a interacção entre elas;

Que tipos de interacção são permitidos;

A forma como são tratadas as solicitações e dadas as respostas pelas várias entidades que interagem dentro dos limites do protocolo;

O que fazer quando é recebida uma mensagem de protocolo inválida;

Quais os formatos próprios na definição das mensagens;

As regras sobre o comportamento dos dados (caso sejam aceites, não aceites ou preferenciais).

Podem existir casos onde um protocolo não especifica algumas regras, por não estar completamente especificado, ou por deliberadamente deixar algumas especificações para outros processos controlados por outro protocolo. Mesmo assim, todas as comunicações exigem um conjunto de regras, de forma a garantir que os dados são transmitidos e recebidos. (Kurose & Ross, 2004)

Em seguida será dada um exemplo de um protocolo numa ligação *TCP (Transmission Control Protocol)*, de forma a ilustrá-lo e a facilitar a sua compreensão. Neste caso o protocolo especifica 3 fases:

- Estabelecimento de ligação;
- Transferência;
- Fecho de ligação.

No caso do estabelecimento da ligação, este é realizado em três passos e no caso do término são necessários quatro passos. No *TCP*, tipicamente existe um servidor (fornece um *socket* e aguarda ligações) e um cliente. (Loshin, 2004)

O cliente inicia a ligação enviando um pacote *TCP*, com uma etiqueta *SYN (Synchronize)* activa, e aguarda que o servidor a aceite, retornando um pacote *SYN+ACK (Synchronize Acknowledge)*. Caso isto não se verifique durante um determinado período de tempo, ocorre um *timeout*. Caso a ligação seja aceite, o cliente retorna um pacote *ACK (Acknowledgement Packet)*. (Loshin, 2004)

Neste processo de trocas, são trocados entre o cliente e o servidor números de sequência iniciais, que servirão para reconhecer os dados, bem como servir de contador de *bytes* durante esta fase. (Loshin, 2004)

No fim desta fase, o servidor inscreve a ligação do cliente numa tabela própria, que contém um número limite de ligações aceites pelo servidor, podendo-se verificar um atraso, como ilustrado na Figura 3.2.

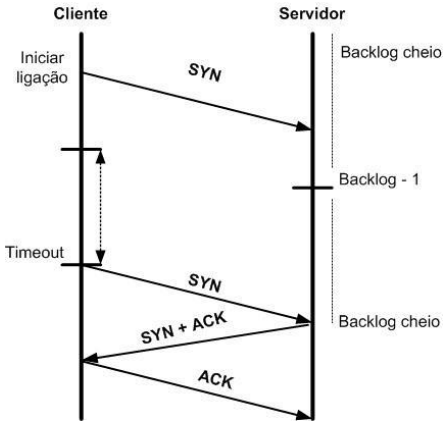

Figura 3.2 - Atraso e time out (Fonte: http://www.tiosam.org/enciclopedia/index.asp?q=TCP)

Na fase de transferência (sessão), o *TCP* possui vários mecanismos que asseguram a fiabilidade e robustez da transmissão. São eles: números de sequência, que garantem a entrega ordenada; o *checksum*, usado para detecção de falhas em segmentos específicos; a confirmação de recepção; e temporizadores que ajustam e contornam eventuais atrasos e perdas de segmentos. (Loshin, 2004) No cabeçalho *TCP* existe sempre um par de números de sequência, doravante referidos como número de sequência e número de confirmação (*ACK*). O emissor determina o seu próprio número de sequência e o receptor confirma o segmento, usando como número *ACK* o número de sequência do emissor. De forma a manter a fiabilidade, o destinatário assegura que recebeu um determinado número de *bytes* contíguos dos segmentos designados. (Loshin, 2004)

Um dos melhoramentos apresentados no *TCP*, foi a eventualidade do receptor aceitar blocos fora da ordem prevista. Esta particularidade designa-se por *selective ACK*, ou apenas *SACK (Selective ACK)*. A montagem ordenada dos segmentos é feita usando os números de sequência, de 32 *bit*, que recomeçam a zero quando superam o valor máximo, ficando com o valor da diferença. O campo *checksum* atesta a integridade do segmento. Esta técnica, apesar de muito inferior a outras técnicas de detecção, como o *CRC (Cyclic Redundancy Check)*, é usada para testes de integridade ao nível da camada 2. (Loshin, 2004)

Contudo, este campo não fica redundante, uma vez que estudos de tráfego revelam que a introdução de erro é bastante frequente entre saltos protegidos por *CRC*, e que este campo detecta a maioria desses erros. No caso da confirmação de recepção dos pacotes *ACK*, esta dá ao emissor a oportunidade para determinar as condições da rede. O cabeçalho *TCP* possui um parâmetro que permite indicar o espaço livre actual do receptor. Assim, o emissor sabe desta forma que só poderá ter em circulação uma determinada quantidade de informação e que terá de esperar pela confirmação de um pacote *ACK* para proceder à actualização da janela. (Loshin, 2004)

Todavia, devido à dimensão do campo, que não pode ser alargado, os limites aparentes da janela vão entre 2 e 65535, o que é muito pouco em redes de alto débito. De forma a evitar esta limitação, é utilizada uma opção especial que autoriza obter múltiplos do valor da janela,

o *TCP window scale*. Este valor aponta quantas vezes o valor da janela, de 16 *bit*, deverá ser executado, por deslocamento de *bits* (para a esquerda) para obter os múltiplos, variando entre 0 e 14. Assim, é possível obter janelas de 1 *GBit*. O parâmetro de escala é determinado somente durante a criação da ligação. (Loshin, 2004)

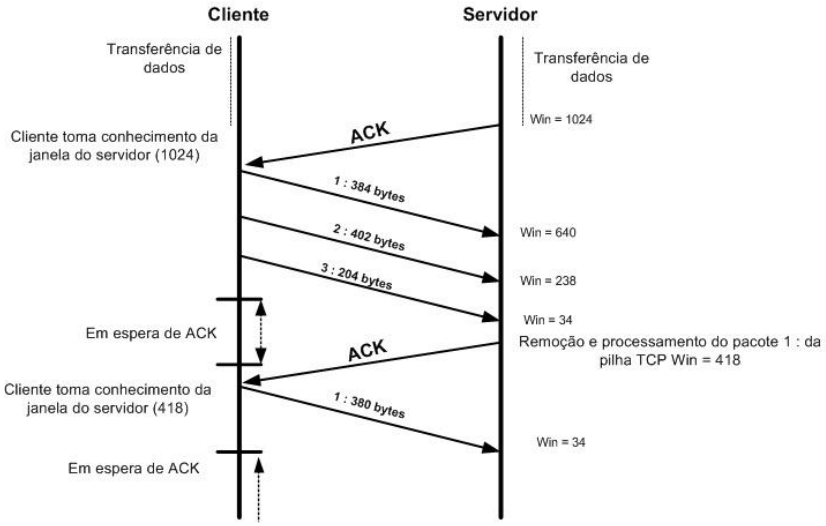

Figura 3.3 - Visualização do ACK recebido (Fonte: http://www.tiosam.org/enciclopedia/index.asp?q=TCP)

A fase de fim da sessão TCP é um processo que se desenrola em quatro fases, em que cada um se responsabiliza pelo fecho da ligação. Sempre que um deles deseja acabar a sessão, envia um pacote com a etiqueta FIN activa. Logo após o envio deve receber uma resposta ACK. Por sua vez, a outra parte procederá da mesma forma, enviando um FIN e deverá responder com um ACK. (Loshin, 2004)

Poderá acontecer que um dos lados não termine a sua sessão. Estes casos são chamados de conexão semi-aberta. A parte que não encerrou a sessão continuará a enviar informação pela ligação, mas isso não se verificará no outro lado. (Loshin, 2004)

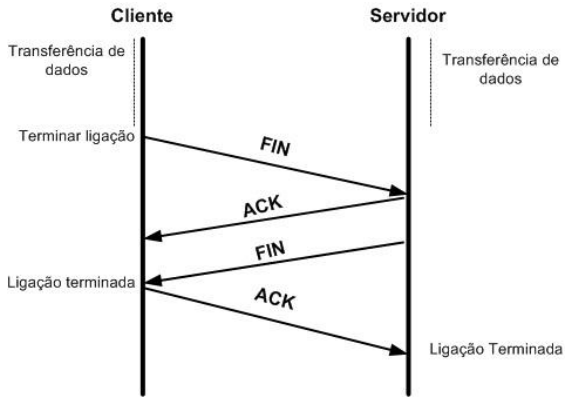

Figura 3.4 - Fim da ligação (Fonte: http://www.tiosam.org/enciclopedia/index.asp?q=TCP)

3.3 Introdução à pilha TCP/IP

Neste ponto são abordados os conceitos referentes à pilha protocolar *TCP/IP* e os protocolos de comunicação e as estruturas de funcionamento. O protocolo *TCP/IP* nasceu na década de 70, inicialmente patrocinado pela *DARPA (Defense Advanced Research Project Agency)*, com o objectivo de poder ser utilizado em redes de computadores. Após a especificação dos protocolos *TCP* e *IP (Internet Protocol)*, o IP foi desenvolvido para ser o protocolo primário da camada de rede. Várias foram as redes do departamento de defesa americano que começaram a ser interligadas. Com estas tecnologias, a rede era conhecida por *ARPANET (Advanced Research Projects Agency Network)*. Mais tarde, deixou de ser apenas para uso militar e começou a expandir-se para instituições governamentais, comerciais e educacionais. (Casad, 2001)

Com o início da década de 80, a combinação entre os protocolos de transporte *TCP* e de rede *IP*, foram adoptados como normas da *ARPANET* e todas as demais redes que se foram interligando a esta tinha como requisito suportar o *TCP/IP*. Hoje em dia, esta nova rede é conhecida como *Internet*, e o número dos seus utilizadores cresceu em todo o mundo. Uma das principais razões para o sucesso do *TCP/IP* foi a especificação dos protocolos, feita através de documentos chamados de *RFC (Request for Comments)*. Para que qualquer protocolo seja adoptado como uma norma *Internet*, é necessária a publicação do seu *RFC*, podendo existir *RFCs* que actuem como sugestões ao anterior. (Casad, 2001)

Assegurada a sua estabilidade, um qualquer membro da *IAB (Internet Architecture Board)* pode sugerir ao comité da *IAB,* que o protocolo seja adoptado como padrão. Para que isso aconteça, um *RFC* é publicado, e, caso não haja objecções durante um determinado período, definido sempre pela entidade, o IAB declara o protocolo como uma norma. Na composição actual da pilha *TCP/IP* participam um grande número de protocolos, e não apenas os dois que lhe deram o nome. Dentro deste conjunto de protocolos podemos enumerar os seguintes: (Casad, 2001)

- ARP (Address Resolution Protocol)
- DHCP (Dynamic Host Configuration Protocol)
- ICMP (Internet Control Message Protocol)
- UDP (User Datagram Protocol)

O *TCP/IP* é um protocolo de sistema e suporta a comunicação numa rede. Uma rede é um conjunto de equipamentos que comunicam entre si num meio de transmissão comum, como se pode ver na Figura 3.5.

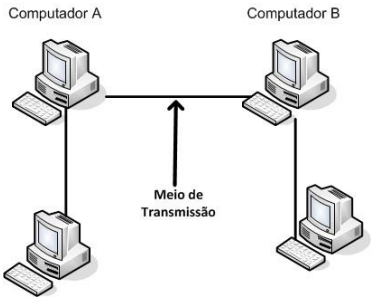

Figura 3.5 - Exemplo básico de uma rede de computadores (Casad, 2001)

Numa rede, os pedidos e os dados passam de um computador para o outro através de um meio de transmissão, que pode ser um cabo de rede ou um cabo telefónico (por exemplo). Na Figura 3.6 o computador **A** deverá conseguir enviar uma mensagem ou um pedido para o computador **B**. O computador **B** terá de entender a mensagem proveniente do computador **A**, e responder-lhe. Um computador interage com o mundo através de uma ou mais aplicações que possibilitam as tarefas e a gestão de entradas e de saídas. Caso o computador seja parte de uma rede, algumas dessas aplicações devem ser capazes de comunicar com outras aplicações que estejam instaladas noutros computadores através da rede. (Casad, 2001)

Um protocolo de rede é um sistema de regras que ajudam a definir o processo complexo de transferência de dados. Os dados viajam de uma aplicação localizada num computador, utilizando o *hardware* de rede, através de um meio de transmissão, até chegarem ao destino correcto. Esses dados são recepcionados pelo *hardware* de rede do computador de destino e posteriormente lidos pela aplicação de destino, como se pode ver na Figura 3.6. (Casad, 2001)

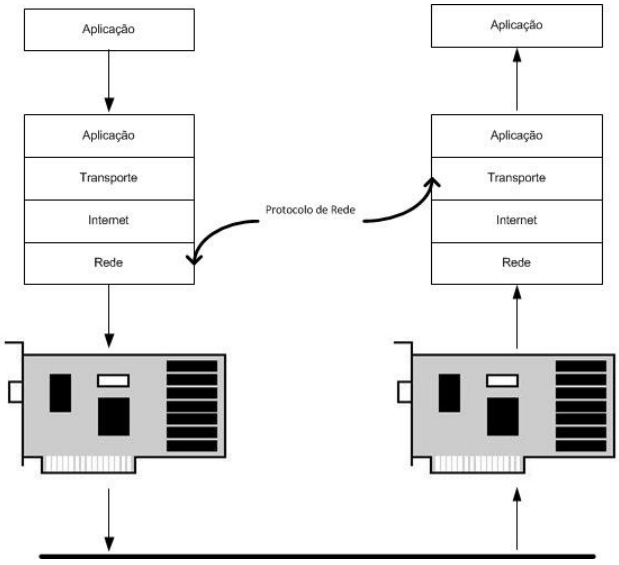

Figura 3.6 - Esquema de funcionamento de um protocolo de rede (Casad, 2001)

O protocolo *TCP/IP* define um processo de comunicação em rede, ou seja, define como uma unidade de dados deve ser e que informação deve conter, para que a máquina que recebe esta informação a possa interpretar de uma forma correcta. O *TCP/IP* forma um sistema completo que define como os dados devem ser processados, transmitidos e recebidos numa rede *TCP/IP*. Na arquitectura de comunicação *TCP/IP* é incluído um conjunto de protocolos de suporte à transmissão de dados. O facto de ser uma arquitectura aberta torna o *TCP/IP* na arquitectura utilizada na maioria das redes de comunicação de dados, incluindo a rede *Internet*. (Casad, 2001)

O *TCP/IP* usa uma estrutura de camadas bastante próximas, como se pode ver na Figura 3.7.

Figura 3.7 - Modelo *TCP/IP* (Casad, 2001)

A camada de aplicação no modelo *TCP/IP* é, em geral, comparada às três camadas superiores do modelo *OSI*. Assim os serviços de compressão, de encriptação, de administração de sessões entre outros, são todos realizados na camada de aplicação. As camadas de transporte, de *Internet*, de acesso à rede e física têm serviços idênticos às camadas de transporte, de rede, de ligação à rede e física do modelo *OSI*, respectivamente. Certos autores optam por juntar as camadas de acesso à rede e física, criando um modelo *TCP/IP* com quatro camadas. (Casad, 2001)

Esta união deve-se ao facto do modelo considerar que existem muitas dependências entre as duas camadas e, consequentemente, englobam muitos dos seus serviços numa só. Por exemplo, quando se alteram as características do meio físico é muito provável que também se tenha de alterar a camada de acesso à rede. Por outro lado, a abordagem com cinco camadas tem a vantagem de que a camada *Internet* passa a ser identificada como uma camada nível 3, correspondendo assim ao mesmo nível da camada de rede no modelo *OSI*. Desta forma, quando se diz que um equipamento implementa serviço de nível 3, não surge qualquer dúvida de interpretação, pois quer no *TCP/IP* quer no *OSI* o nível 3 tem as mesmas funções. (Casad, 2001)

A arquitectura *TCP/IP* é hoje formada por um vasto conjunto de protocolos que se distribuem pelas diferentes camadas do modelo, na Figura 3.8 é exemplificado um conjunto de protocolos usados no *TCP/IP*.

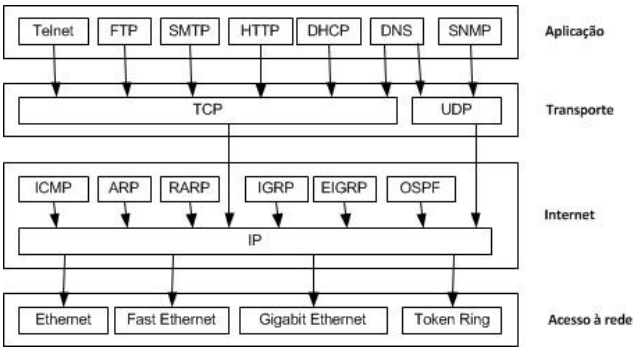

Figura 3.8 - Protocolos *TCP/IP*

O termo pilha de protocolos define um grupo de protocolos que são usados em conjunto. Os protocolos de comunicação são um conjunto de regras que tem como objectivo permitir a comunicação entre máquinas. Como esse conjunto de regras é extenso, essas regras (protocolos de comunicação) são agrupadas em níveis hierárquicos. A pilha *TCP/IP* é dividida em cinco camadas, como se pode ver na Figura 3.9. Cada uma tem uma responsabilidade específica, que torna a complexidade de implementação do protocolo mais reduzida. Cada camada executa um conjunto de funções com a finalidade de providenciar serviços para a camada que está imediatamente acima. O fluxo de dados é realizado da seguinte forma: (Casad, 2001)

1. Em primeiro, a camada de transporte recebe dados da camada de aplicação, e de seguida adiciona um cabeçalho do protocolo de transmissão e passa os dados para a camada de rede;

2. Na camada de rede é adicionado o cabeçalho do protocolo de rede e o pacote passa para a camada de ligação de dados;

3. A camada de ligação de dados também adiciona um cabeçalho antes de enviar os dados para a camada física;

4. Esta vai ser responsável pela transmissão desses dados, sob a forma de bits, através do meio de comunicação.

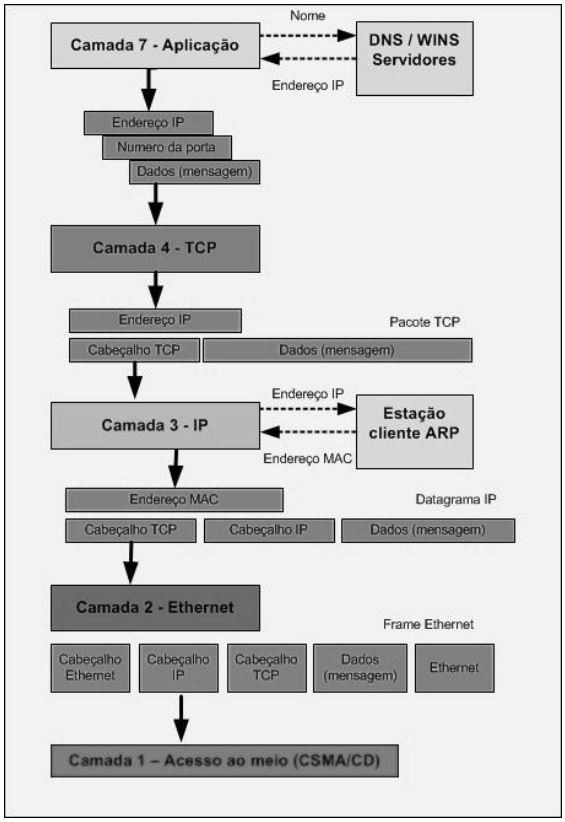

Figura 3.9 - Pilha TCP/IP (Sicero, 2004)

O termo *Ethernet* refere-se ao padrão publicado no início da década de 80 pelas empresas *Digital*, *Intel* e *Xerox*. Nos dias de hoje esta tecnologia foi adoptada como padrão tecnológico nas redes locais *(LAN)*, tornando-se assim a mais utilizada no mundo em conjunto com o *TCP/IP*. O método de acesso ao meio utilizado por esta tecnologia é o *CSMA/CD (Carrier Sense Multiple Access with Collision Detection)*, usando um endereçamento de 48 *bits*. É usado como referência ao padrão *IEEE (Institute of Electrical and Electronics Engineers)* 802.3, o qual se preocupa com elementos da camada física e de ligação de dados. (Sicero, 2004)

Ao nível da camada de ligação de dados existem subcamadas de controlo lógico de ligação, designadas por *LLC (Logical Link Control)* e *MAC (Media Access Control)*. Enquanto a camada *LLC* é responsável por identificar os dados transportados, o *MAC* é responsável pelo protocolo que tem como função arbitrar o acesso ao sistema *Ethernet*. O núcleo do sistema *Ethernet* é a trama. O *hardware* de rede *(NIC (Network Interface Card)*, cabos, entre outros*)* existe e tem como função movimentar tramas entre equipamentos. Os *bits* da trama *Ethernet* são distribuídos em campos específicos, como mostra a Figura 3.10, onde se pode ver a distribuição dos campos na trama. (Sicero, 2004)

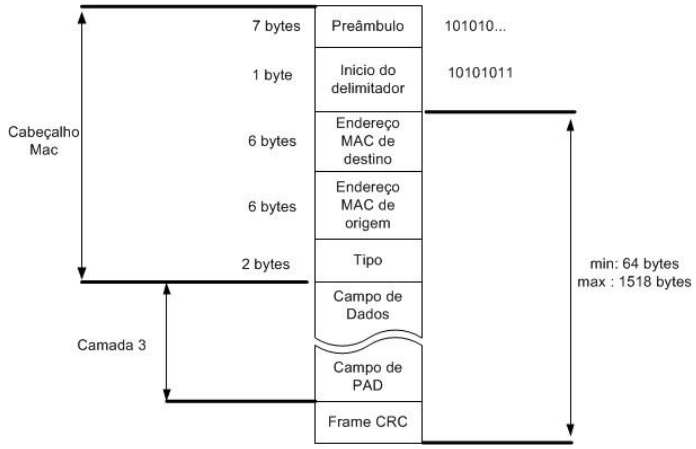

Figura 3.10 - Trama *Ethernet* (Sicero, 2004)

Na trama, o primeiro campo é o preâmbulo, que é responsável pela sincronização do receptor com o sinal, actuando como um aviso de que uma trama está a ser transmitida. A sua composição é uma sequência de 64 *bits* de zeros e uns alternados. Na sequência aparece os endereços físicos de destino e do emissor, ocupando cada um 48 *bits*. É da responsabilidade do protocolo *ARP* a tradução destes endereços físicos em endereços *IP*. (Sicero, 2004)

Os próximos dois *bytes* identificam o tipo dos dados que está encapsulado pela trama, seguindo-se os dados que vão ser transmitidos e que podem ser um pacote do tipo *IP*, *ARP*, entre outros. No fim da trama vem o *CRC (Cyclic Redundancy Check)* que tem por finalidade detectar erros na trama transmitida. Existe um tamanho mínimo para a trama, sendo necessários pelo menos 46 *bytes* no campo dos dados, podendo, caso seja necessário, ser acrescentados *bytes* adicionais, de forma a atingir o tamanho mínimo. (Sicero, 2004)

3.3.1.1 ARP

O protocolo *ARP* tem por finalidade resolver o problema da associação entre endereços da camada de ligação de dados e da camada de rede, estando especificado no *RFC* 826. Quando uma trama *Ethernet* é enviada de um posto para outro, é o endereço *Ethernet* (48 *bits*) que identifica a interface que deve receber a trama, e não o endereço de destino que está no cabeçalho do protocolo da camada superior. Assim sendo, a finalidade do protocolo é a de converter endereços de protocolos (endereços *IP*) para cabeçalhos de redes locais (endereços *Ethernet*). (Sicero, 2004)

Na Figura 3.11 são exemplificados os campos de cabeçalho, em que os dois primeiros (16 *bits*) identificam os vários tipos de endereços e a conversão realizada. Seguem-se dois campos de 8 *bits*, que determinam o tamanho em *bytes* dos campos de endereço a ser convertidos. Os quatro primeiros campos garantem a flexibilidade do protocolo, pois permitem que qualquer endereço de protocolo seja convertido num qualquer tipo de endereço de *hardware*. Por fim, aparecem dois campos que devem ser completados com endereços de *hardware* e protocolo do emissor e, de seguida, vêm os endereços de *hardware* e *software* do destinatário da mensagem. (Sicero, 2004)

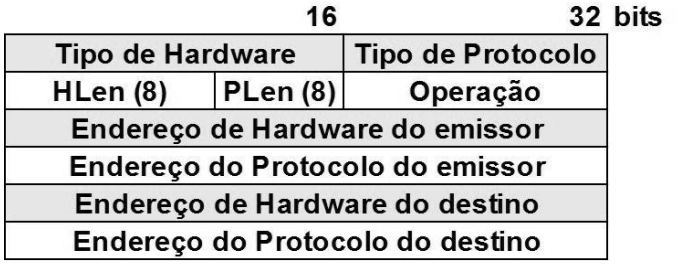

Figura 3.11 - Cabeçalho *ARP* (Sicero, 2004)

3.3.1.2 IP

O protocolo *IP* é o coração da pilha *TCP/IP*. O *IP* fornece um serviço de entrega de datagrama sem ligação pré-estabelecida e sem garantia, pelo que não é garantida a entrega do pacote, ou seja, providencia um serviço de melhor esforço (*best effort*). Sempre que há uma falha, o *IP* usa o seu mecanismo simples de manipulação de erros, descartando o datagrama e procedendo ao envio de uma mensagem *ICMP* de volta para o emissor. Como não mantém uma ligação, a manipulação de cada datagrama é realizada de forma individual dos restantes, podendo ocorrer uma chegada desorganizada dos pacotes ao destinatário. Caso se verifique a necessidade de fiabilidade na entrega, deverá ser utilizado um protocolo de uma das camadas superiores. (Sicero, 2004)

Na Figura 3.12 pode ser observado o cabeçalho *IP*. O primeiro campo (4 *bits*) é para a identificação da versão do protocolo. O segundo identifica o tamanho em unidades de 4 *bytes*. De seguida aparece o campo tipo de serviço, sendo, neste caso, ignorados os primeiros 3 *bits*, enquanto os restantes quatro sinalizam o atraso mínimo, o débito máximo (*throughput*), a máxima fiabilidade e o mínimo custo. O último *bit* deste campo não deverá ser usado, sendo o seu valor 0. (Sicero, 2004)

A informação sobre o tamanho completo do datagrama *IP* em *bytes* é fornecida pelo campo "tamanho total", o qual, conjuntamente com o campo "tamanho do cabeçalho", permite saber com exactidão onde se inicia o campo de dados dentro do datagrama *IP*. O campo "identificação" identifica os datagramas enviados por determinado posto. Por norma, a cada envio, este número é incrementado em um. Os campos etiquetas e fragmento *offset* são usados durante a fragmentação e reconstrução dos datagramas. O campo *TTL (Time to Live)* especifica o número máximo de *routers* pelos quais o datagrama pode passar, limitando

desta forma o tempo de vida do pacote. Sempre que passa por um *router* esse valor é diminuído, sendo o pacote descartado quando o seu valor atinge 0 e enviada uma mensagem *ICMP* ao emissor do datagrama. Este mecanismo actua de forma preventiva com o objectivo de evitar que um pacote fique retido entre dois *routers* num ciclo infinito. (Sicero, 2004)

O campo protocolo identifica o tipo de dados que estão encapsulados no datagrama *IP*, sendo usado para desmultiplexar os pacotes recebidos e identificar para qual protocolo o *IP* deve passar o pacote. O campo seguinte armazena o valor do *checksum* que é executado somente no cabeçalho do datagrama. Em seguida surgem os campos com os endereços da fonte e do destino da mensagem, os quais são endereços de 32 *bits*, seguidos pelos campos com as opções do protocolo *IP*, que podem ser referentes a segurança, opções de encaminhamento e controlo de tempo, sendo que já estão em desuso e só uma minoria das implementações é que garantem o seu suporte. (Sicero, 2004)

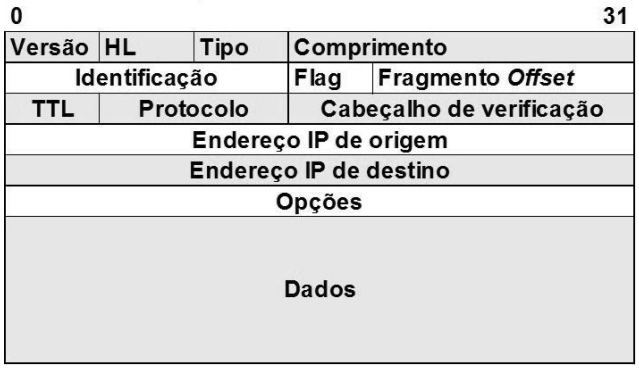

Figura 3.12 - Cabeçalho *IP* (Sicero, 2004)

3.3.1.3 ICMP

O protocolo *ICMP* é considerado parte da camada *IP*, comunicando mensagens de erro que requerem atenção. As mensagens *ICMP* são produzidas pela camada *IP* ou por um protocolo de mais alto nível. Algumas mensagens de *ICMP* provocam erros, os quais devem ser comunicados aos processos de utilizadores. As mensagens *ICMP* são definidas por um tipo e um código, existindo um grande número de tipos e códigos, sendo assim a solução para a detecção de erros no envio de pacotes *IP*. (Sicero, 2004)

3.3.1.4 UDP

O protocolo *UDP* é um simples protocolo de transporte orientado a datagramas. Toda a operação de envio produz um datagrama *UDP*, que causa a geração também de um datagrama *IP* para ser enviado. No protocolo *UDP* não existe a garantia de que os datagramas chegarão ao seu destino. Caso a quantidade de informação do datagrama a ser enviada seja superior ao *MTU (Maximum Transport Unit)*, o datagrama *IP* é fragmentado. (Sicero, 2004)

O cabeçalho *UDP* pode ser visto na Figura 3.13. O campo "porta de origem" identifica o processo que enviou a mensagem, e o campo "porta de destino" identifica o processo que

deve receber a mensagem. O campo “comprimento” dá informação sobre o tamanho do datagrama *UDP* (cabeçalho + dados) em *bytes*. O valor mínimo deste campo é de 8, ou seja, é possível enviar um datagrama *UDP* com 0 *bytes* de dados. O próximo campo contém o valor de *checksum* e é calculado nos campos do cabeçalho e de dados. (Sicero, 2004)

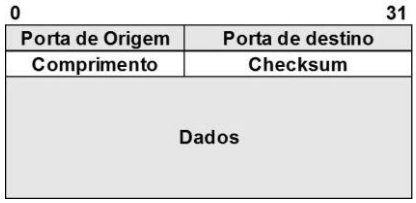

Figura 3.13 - Cabeçalho *UDP* (Sicero, 2004)

Nos seguintes pontos irá ser abordado o *TCP/IP* com mais detalhe ao nível das suas camadas de aplicação, transporte, rede e ligação de dados.

3.3.2 Camada de aplicação

Nos meados da década de 70 começaram a ser criadas as primeiras aplicações que trabalhavam em rede. Na década seguinte estas aplicações começaram a ganhar popularidade destacando-se o correio electrónico, o acesso a computadores remotos, a transferência de arquivos, a conversação interactiva e, claro, mais tarde, com o aparecimento em meados da década de 90, o acesso massificado à *Web*. Com este acesso apareceram aplicações como vídeo em tempo real, rádio, telefonia *IP*, videoconferência e outras mais. (Casad, 2001)

Assim são importantes na camada de aplicação os conceitos: (Casad, 2001)

- Cliente/Servidor;
- Processos;
- Sockets;
- Interfaces da camada de transporte.

A camada de aplicação é um bom começo para iniciar o estudo de protocolos, pois dela fazem parte diversas aplicações que se utilizam no dia-a-dia, e será um bom ponto de partida para as restantes camadas: transporte, rede e física, que são abordadas nos próximos pontos. O cerne do desenvolvimento de qualquer aplicação de rede é escrever um programa que corra num sistema final e os programas comuniquem entre eles. Por exemplo, na *Web* existem dois programas distintos, um que corre no navegador *Web* do computador cliente e o outro que corre num servidor, podendo ou não existir semelhanças entre os dois. Quando uma pessoa desenvolve uma nova aplicação para ambiente Web deverá ter em consideração que essa aplicação pode correr em ambientes distintos. De seguida será abordado o tema de arquitecturas de aplicação de rede. (Kurose & Ross, 2004)

3.3.2.1 Arquitecturas de aplicação de rede

A arquitectura da aplicação é um ponto que deve ser bem equacionado no desenvolvimento da aplicação, pois quando se faz a escolha da arquitectura a pessoa que irá desenvolver a aplicação pode optar por uma das seguintes escolhas: (Kurose & Ross, 2004)

- Cliente-Servidor;
- P2P;
- Híbrida Cliente-Servidor/*P2P*.

Na arquitectura cliente-servidor existe uma máquina sempre em funcionamento (servidor), que recebe pedidos de várias máquinas cliente, que nem sempre estão em funcionamento. Um exemplo clássico é o de uma aplicação *Web*, em que o servidor *Web* está sempre disponível, em condições normais, para receber pedidos dos clientes. O servidor quando recebe um pedido de um objecto de um qualquer cliente, responde enviando o objecto requisitado. (Kurose & Ross, 2004)

Na arquitectura cliente-servidor, os clientes não comunicam directamente entre si. Por exemplo, dois navegadores *Web* não comunicam directamente. Uma característica de arquitectura cliente-servidor é que o servidor tem um endereço fixo, que é bem conhecido por parte dos clientes. Por isso, o servidor está sempre visível para o contacto feito pelos clientes. Na Figura 3.14 podemos ver um exemplo de duas aplicações. (Kurose & Ross, 2004)

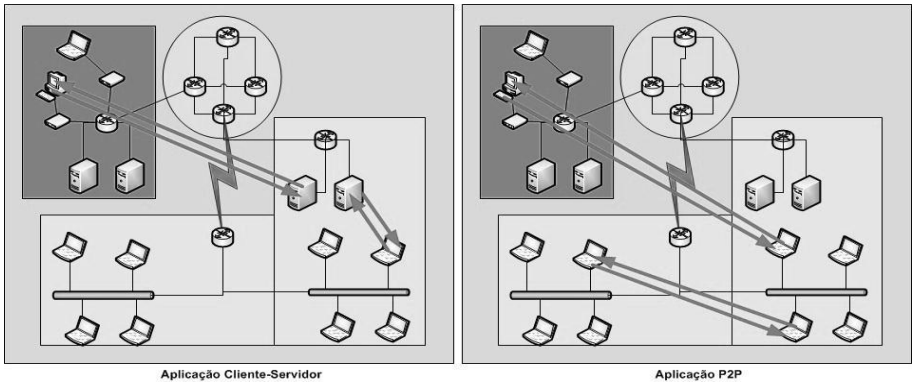

Figura 3.14 - Cliente/Servidor e P2P (Kurose & Ross, 2004)

Na arquitectura *P2P* pura, não existe um servidor que esteja sempre em funcionamento no núcleo da aplicação. Em vez disso existem pares arbitrários de máquinas, que são denominados de *peers*, que comunicam directamente entre si. Uma vez que a comunicação é directa, esta arquitectura é denominada *peer-to-peer*. As máquinas envolvidas neste tipo de arquitecturas não necessitam de estar continuamente ligadas e podem variar os seus endereços *IP* sempre que se ligarem. Um exemplo de uma aplicação *P2P* pura é a *Gnutella*, uma aplicação *P2P* em código-fonte aberto. Nesta aplicação qualquer máquina pode requisitar ou enviar arquivos, consultar e responder. (Kurose & Ross, 2004)

Uma das características mais fortes desta arquitectura é a sua escalabilidade, podendo cada máquina funcionar como um servidor que partilha dados, contribuindo com recursos. O reverso da medalha é que dada esta escalabilidade possível, pode-se tornar impossível administrar este tipo de aplicações. Ambas as arquitecturas aqui referidas são comuns nas aplicações de rede, mas uma grande parte delas estão organizadas como arquitecturas híbridas cliente-servidor/*P2P*. Por exemplo, o *Napster* foi uma aplicação *P2P* que tinha por base a partilha entre *peers* sem recorrer a servidores dedicados, mas também tinha arquitectura cliente-servidor, pois o *peer* consultava um servidor onde eram determinados quais os *peers* que tinham a informação pretendida. (Kurose & Ross, 2004)

A camada de aplicação é a camada que permite a comunicação com outros programas e que a grande parte dos programas de rede utiliza. Os processos que são executados nesta camada são específicos da aplicação. Os objectos são passados do programa de rede, no formato usado internamente por essa aplicação, para serem codificados dentro do padrão de um protocolo. Alguns programas específicos são levados em conta nessa camada. Eles fornecem serviços que suportam directamente aplicações do utilizador. Os seus correspondentes protocolos incluem o *HTTP*, *FTP*, *SMTP*, *SSH*, *DNS* e muitos outros, como pode ser visualizado na Figura 3.15. Uma vez que o objecto de uma aplicação foi codificado dentro de um padrão de um protocolo da camada de aplicação, ele será passado para a próxima camada da pilha *IP*. (Kurose & Ross, 2004)

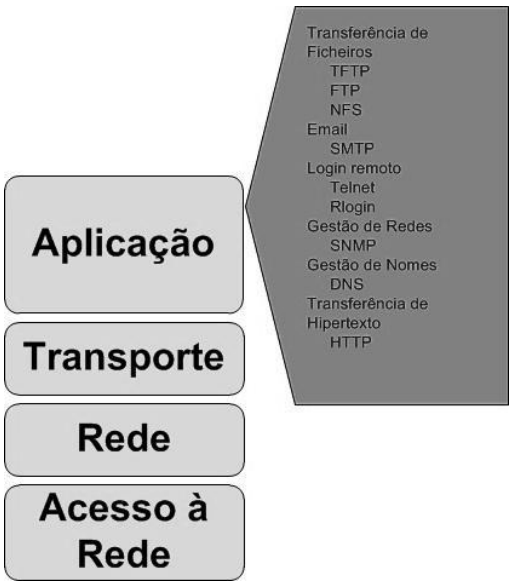

Figura 3.15 - Camada de Aplicação (Fonte: http://ladingmerah.blogspot.pt/2009_03_29_archive.html)

3.3.3 Camada de transporte

Esta camada encontra-se posicionada entre a camada de aplicação e a camada de rede, desempenhando um papel central nesta arquitectura, pois é responsável pelo fornecimento de serviços de comunicação directamente aos processos da aplicação que são executados em máquinas diferentes. É nesta camada que aparecem os protocolos *TCP* e *UDP*. É de salientar a relação entre a camada de transporte e a camada de rede. Para realizar um serviço de entrega entre sistemas finais é necessário utilizar um protocolo não orientado à conexão o *UDP*. (Kurose & Ross, 2004)

Um dos principais problemas das redes de computadores é o estabelecimento entre duas entidades de uma comunicação de forma confiável por um meio onde pode haver perda e corrupção de dados, problema que pode ser resolvido utilizando o *TCP*, pois garante controlo de erros, controlo de fluxo e verificação dos dados enviados, garantido o envio e recepção dos dados entre o emissor e receptor. Um outro problema é o controlo da taxa de transmissão, de forma a evitar-se o congestionamento da rede. Um protocolo da camada de transporte

fornece comunicação lógica entre processos da aplicação que são executados em máquinas diferentes. (Kurose & Ross, 2004)

A comunicação lógica nesse contexto significa, que do ponto de vista de uma qualquer aplicação, tudo ocorre como se as máquinas finais estivessem a executar os processos estando directamente ligadas, embora, na verdade, essas máquinas possam estar em pontos completamente opostos no planeta, comunicando através de uma variedade enorme de *routers* e de diferentes tipos de ligações. Os processos de aplicação utilizam uma comunicação lógica fornecida pela camada de transporte que possibilita o envio de mensagens entre si, livres de preocupações dos detalhes da infra-estrutura física que utilizam, como se pode ver na Figura 3.16, que ilustra uma comunicação lógica. (Kurose & Ross, 2004)

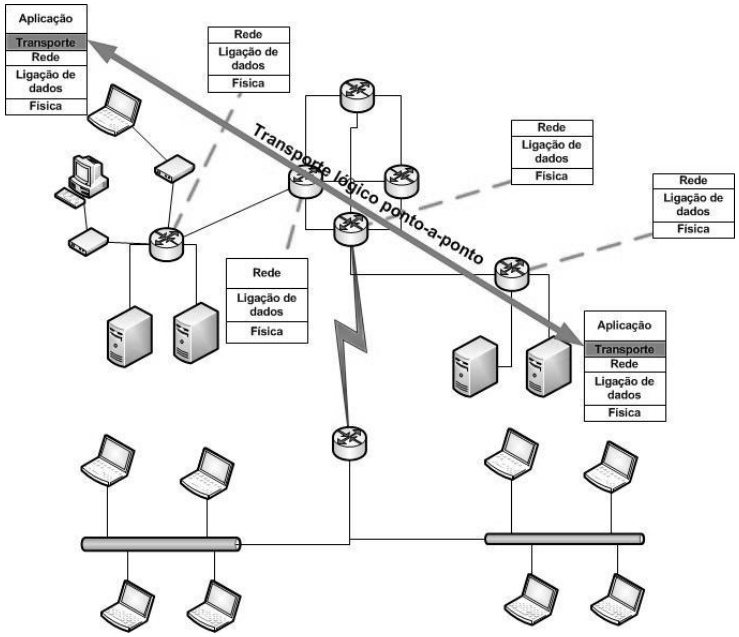

Figura 3.16 - Comunicação Lógica (Kurose & Ross, 2004)

Na Figura 3.16 pode-se ver que os protocolos da camada de transporte são implementados nos sistemas finais, e não nos equipamentos de rede. No lado do emissor, a camada de transporte converte as mensagens que recebe de um processo da aplicação em pacotes da camada de transporte, chamados de segmentos da camada de transporte. Isto é realizado fragmentando-se as mensagens da aplicação em pedaços menores e adicionando-se-lhe um cabeçalho da camada de transporte a cada parte, de forma a criar o segmento da camada de transporte. (Kurose & Ross, 2004)

Em seguida a camada de transporte passa o segmento para a camada de rede, onde o mesmo é encapsulado num pacote da camada de rede e enviado para o destino. O processo inverte-se no lado do destinatário da mensagem. É importante salientar que os equipamentos de encaminhamento actuam somente nos campos do datagrama na camada de rede, e não examinam os campos do segmento da camada de transporte. A camada de transporte pode ser vista de uma forma mais generalista, onde uma rede *TCP/IP* disponibiliza dois protocolos

de transporte distintos: o *UDP* que fornece à aplicação um serviço não fiável, mais concretamente não orientado à conexão; e o *TCP*, que ao invés do *UDP* dá garantias de um serviço fiável, uma vez que é orientado para a conexão. (Kurose & Ross, 2004)

Quando se projecta uma aplicação de rede, o programador deverá especificar um dos dois protocolos acima referidos. No caso do *TCP* e do *UDP* a responsabilidade é ampliar o serviço de entrega *IP* nos sistemas finais. O processo de entrega entre as máquinas finais é denominado de multiplexação/desmultiplexação da camada de transporte. Ambos os protocolos fornecem a verificação de integridade ao incluir campos de detecção de erros nos cabeçalhos. Os serviços de entrega de dados, processo a processo, e a verificação de erros são os dois únicos serviços que o *UDP* fornece. (Kurose & Ross, 2004)

Assim o *UDP* torna-se num protocolo onde o serviço de entrega não é fiável, pois não garante que os dados enviados por um processo cheguem e quando chegam, de forma intacta, ao processo destinatário. Por outro lado, o *TCP* oferece vários serviços confiáveis, como, por exemplo, a transferência fiável de dados. Para isso usa um controlo de fluxo, números de sequência, reconhecimentos e temporizadores, assegurando, desta forma, que os dados do processo remetente são entregues ao processo de destino de forma correcta. (Kurose & Ross, 2004)

Com esta funcionalidade o *TCP* transforma um serviço não fiável do *IP* entre máquinas finais num serviço fiável de transporte entre processos. O *TCP* fornece o controlo de congestionamento, que é um serviço fornecido à aplicação requerente, um serviço aplicado à Internet para o bom funcionamento geral da rede, pois este evita que qualquer outra conexão *TCP* sobrecarregue as ligações e máquinas que estão a comunicar entre elas com quantidades de tráfego excessivo. (Kurose & Ross, 2004)

Na teoria o *TCP* permite que ligações *TCP* quando estão a gerar tráfego possam usar a mesma ligação de forma semelhante, mesmo que essa ligação esteja congestionada. Este controlo é feito pela regulação da taxa que é permitida para envio por parte do emissor para a rede. Voltando a comparar com o *UDP*, este não regula o tráfego, podendo enviar o tráfego à taxa que pretender e pelo tempo pretendido. Assim sendo, o *TCP* torna-se num protocolo mais complexo. (Kurose & Ross, 2004)

3.3.3.1 Multiplexação / Desmultiplexação

Neste ponto pretende-se dar uma ideia do processo de multiplexação e desmultiplexação na camada de transporte. Na máquina final, a camada de transporte recebe os segmentos da camada de rede logo abaixo, e tem a responsabilidade de os entregar ao processo de aplicação. Na Figura 3.17 descreve-se graficamente o processo. (Kurose & Ross, 2004)

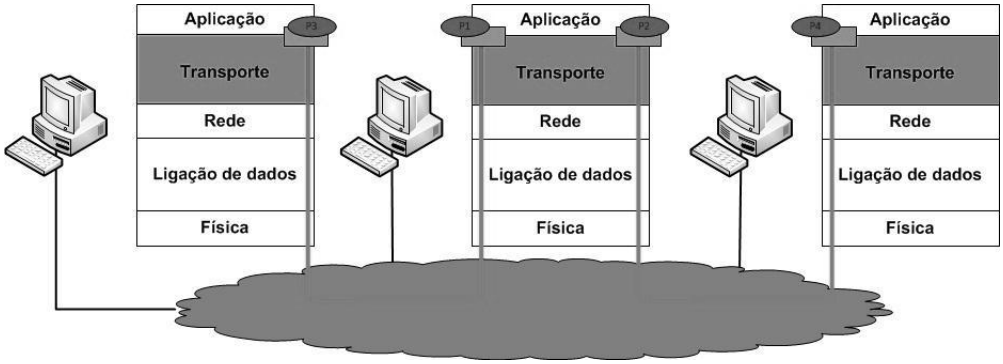

Figura 3.17 - Multiplexação e Desmultiplexação na camada de transporte (Kurose & Ross, 2004)

Como se pode observar na Figura 3.17, a camada de transporte da máquina destinatária, na verdade, não entrega os dados directamente a um processo, mas sim a um *socket* que serve como intermediário. Como pode existir mais do que um *socket* activo na máquina de destino, cada um tem que ter um identificador exclusivo, cujo formato apenas depende se é *socket TCP* ou *UDP*. (Kurose & Ross, 2004)

Se se considerar que a máquina de destino direcciona à porta correcta um segmento da camada de transporte, cada segmento tem um conjunto de campos específicos para essa finalidade. Na máquina receptora a camada de transporte analisa esses campos para identificar a porta receptora e direccionar o segmento para esse *socket*. Este processo de entrega dos dados contidos num segmento da camada de transporte ao respectivo *socket* é chamado de desmultiplexação. (Kurose & Ross, 2004) Por seu lado, o processo de reunir na máquina de origem as porções dos dados provenientes das diversas portas e de encapsular cada porção de dados com informações do cabeçalho, informação que mais tarde será usada no processo de desmultiplexação por forma a criar segmentos e passá-los para a rede, é denominada de multiplexação. No próximo ponto será abordada a camada de rede. (Kurose & Ross, 2004)

3.3.4 Camada de rede

A camada de rede é das mais complexas na pilha de protocolos. Para melhor compreender alguns conceitos é analisado um exemplo de uma rede simples com duas máquinas, **H1** e **H2**, e vários equipamentos *router* no caminho entre **H1** e **H2**, descrito na Figura 3.18. Supondo que **H1** está a enviar informação para **H2**, na máquina **H1** a camada de rede pega nos segmentos provenientes da camada de transporte e encapsula-os num datagrama que é enviado para o seu destino a máquina **H2**. O processo inicia-se com o envio desses datagramas para o equipamento de rede mais próximo que será o *router* **R1**. No ponto de destino da informação, **H2**, a camada de rede recebe os datagramas do seu *router* vizinho, **R2**, o qual irá extrair a informação através do desencapsulamento dos dados e posterior entrega à camada de transporte em **H2**. (Kurose & Ross, 2004)

O papel primordial dos *routers* é o de repassar os datagramas das ligações de entrada para as ligações de saída. Na Figura 3.18, podemos observar que nos *routers* é apenas exibida uma parte da pilha de protocolos, isto é, sem as camadas superiores acima da camada de rede, pois estes equipamentos com a excepção da finalidade de controlo não executam as camadas de aplicação e transporte. (Kurose & Ross, 2004)

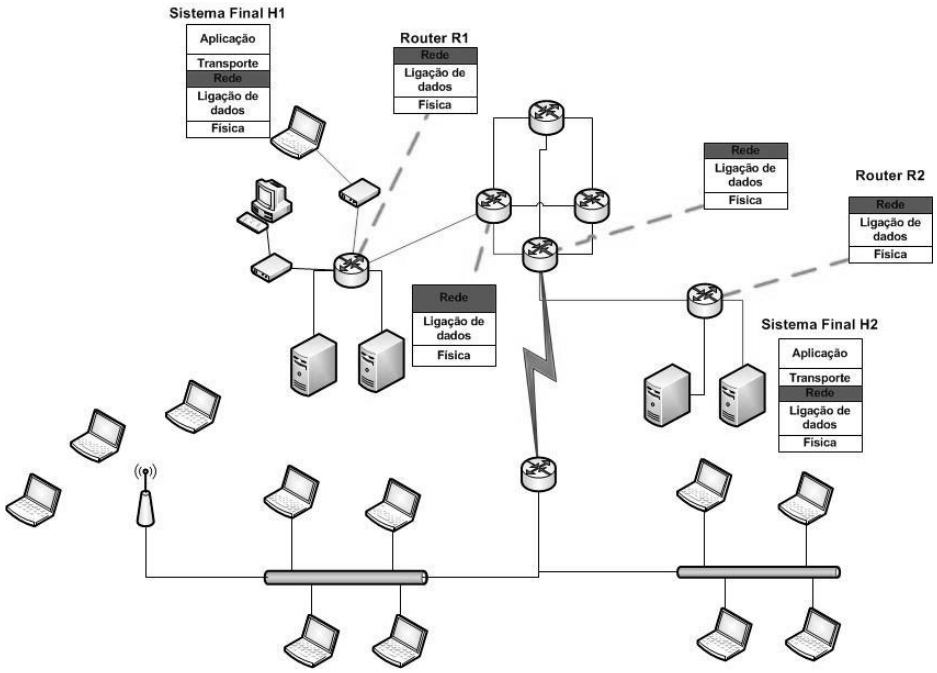

Figura 3.18 - Camada de Rede (Kurose & Ross, 2004)

É importante perceber o conceito das funções de transferência e encaminhamento, pois são os mecanismos que permitem o transporte de informação entre duas máquinas, usando a camada de rede. (Casad, 2001)

Transferência - quando um qualquer pacote chega à ligação de entrada de um *router*, este deve conduzi-lo à ligação de saída.

Encaminhamento - é o processo que a camada de rede utiliza para determinar a rota ou caminho dos pacotes desde o emissor até ao seu destinatário, sendo para isso usados algoritmos de encaminhamento.

Um conceito bastante importante nos equipamentos de encaminhamento é o da tabela de encaminhamento. Esta tabela permite que o *router* transmita o valor de um campo no cabeçalho do pacote que chega, e use esse valor para indexar na sua tabela de encaminhamento. O resultado da tabela de encaminhamento é indicar qual das interfaces de ligação do *router* é usada para transferir o pacote, como pode ser visto na Figura 3.19 que demonstra esse processo. (Kurose & Ross, 2004)

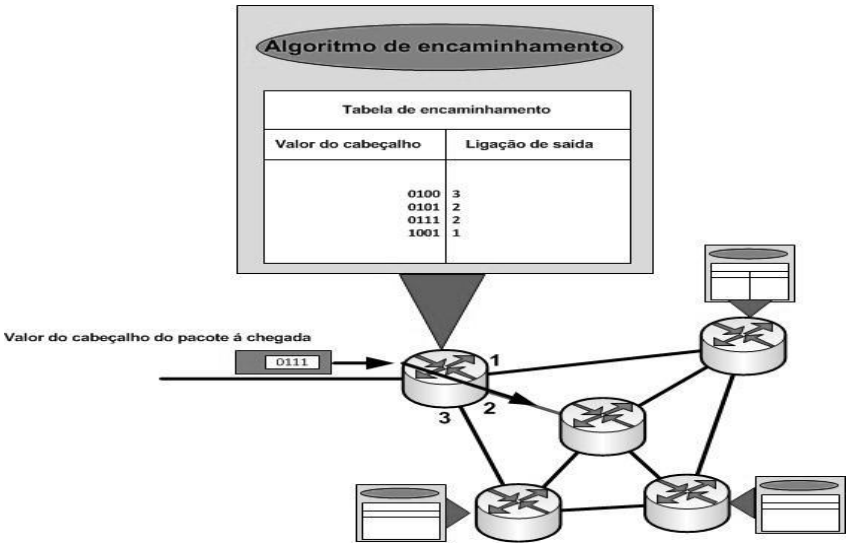

Figura 3.19 - Algoritmo de encaminhamento (Kurose & Ross, 2004)

Uma função também importante é o estabelecimento da conexão antes dos dados fluírem do emissor para o receptor, pois isso permite ao emissor e destinatário o estabelecimento da informação de estado necessária, como, por exemplo, um número de sequência e o tamanho da janela de controlo de fluxo. Desta forma avança-se para a camada física, que será abordada no próximo ponto, não esquecendo os pontos em que foram abordados o *IP* e *ICMP* relevantes para a camada de rede. (Kurose & Ross, 2004)

3.3.5 Camada de Ligação de Dados

No ponto anterior vimos que a camada de rede fornece um serviço de comunicação entre duas máquinas finais. Como se pode ver na Figura 3.20, todo o caminho de comunicação se baseia em ligações de comunicação que tem início na máquina final de origem, passando por uma série de equipamentos (*routers*) até á máquina final de destino. (Kurose & Ross, 2004)

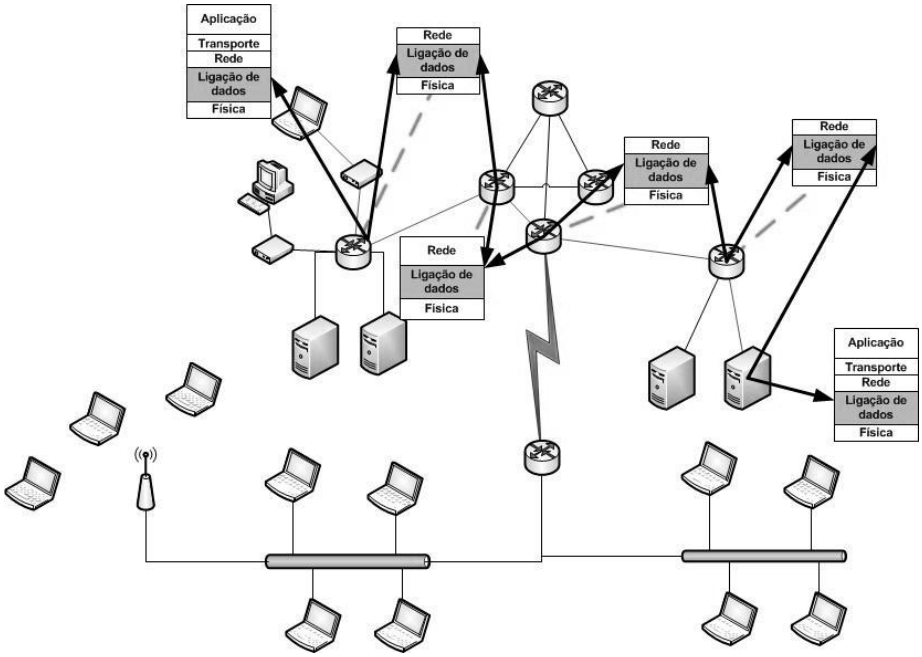

Figura 3.20 - Camada de ligação de dados (Kurose & Ross, 2004)

Dado que continuamos a descer na pilha protocolar TCP/IP até à camada de ligação, é normal que se deduza que os pacotes são enviados pelas ligações individuais dentro do caminho de comunicação, que é resultante do envio de informação da máquina A para a máquina B. No caso do canal de *broadcast*, e dado que existem várias máquinas ligadas no mesmo meio, é necessário um protocolo de acesso ao meio que evite as colisões e que faça a coordenação das transmissões. A coordenação do acesso a uma ligação ponto-a-ponto é trivial, mas existem questões importantes que terão de ser levadas em conta, como o enquadramento, a transferência fiável de dados, a detecção de erros e o controlo de fluxo. O protocolo da camada de ligação de dados define o formato dos pacotes trocados entre as máquinas da extremidade da ligação, bem como as acções realizadas ao enviar e receber os pacotes. (Kurose & Ross, 2004)

As unidades de dados trocados na camada de ligação de dados são chamadas de tramas. Cada trama, por norma, encapsula um datagrama da camada de rede. Entre as acções realizadas pela camada de ligação de dados estão a detecção de erros, retransmissão, controlo de fluxo e acesso aleatório. Como exemplos de protocolos da camada de ligação de dados temos os protocolos *Ethernet*, *token ring* e *PPP (Point-to-Point Protocol)*, entre outros. Assim sendo, a camada de rede tem por tarefa movimentar segmentos da camada de transporte fim-a-fim. Um protocolo da camada de ligação de dados é encarregado de movimentar os datagramas da camada de rede por uma única ligação. (Kurose & Ross, 2004)

A comunicação é realizada por um adaptador, que não é mais que uma placa, por exemplo, um *NIC (Network Interface Card)*. Como demonstrado na Figura 3.21, o datagrama é

encapsulado numa trama e depois transmitido através de uma ligação, sendo que o processo é revertido na outra extremidade.

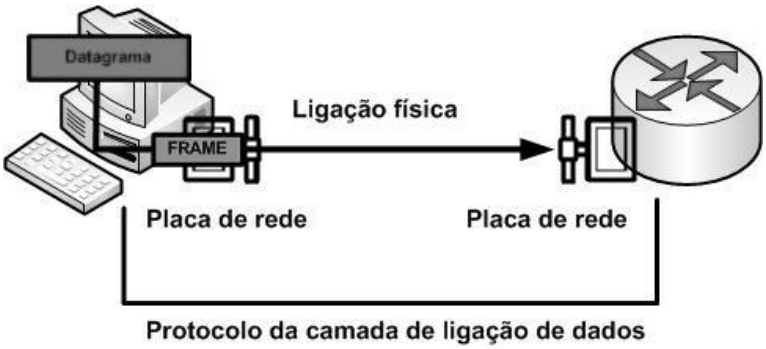

Figura 3.21 - Protocolo da camada ligação de dados (Kurose & Ross, 2004)

3.4 Conclusão

Neste capítulo foram estudadas noções básicas de rede, tendo-se abordado as entidades, os sistemas, a rede e os protocolos. É também mencionado o tema da pilha *TCP/IP* e dos protocolos *ARP, IP, ICMP* e *UDP*. No estudo da pilha protocolar *TCP/IP* foi descrita a camada de aplicação, com uma abordagem às suas arquitecturas de rede. Em seguida foi descrita a camada de transporte, onde foi abordado o tema da multiplexação e desmultiplexação, e por fim as camadas de rede e ligação de dados. No próximo capítulo é abordado o tema da segurança informática.

4 Segurança Informática

4.1 Introdução

Hoje em dia, e com o crescimento dos sistemas de informação, a sua nova estrutura e as características gerais da sociedade actual, funcionando como uma sociedade sempre ligada em rede, a maioria das organizações e pessoas tornaram-se dependentes dos seus sistemas informáticos, de forma a gerir as suas actividades. (Magalhães & Grilo, 2006)

Segundo os autores (Mendyk-Krajewska & Mazur, 2010), estima-se que cerca de 1,5 mil milhões de pessoas usem a Internet dado o seu carácter universal e a panóplia de serviços que esta oferece. Este valor reporta ao ano de 2010, sendo natural que tenha aumentado nos últimos tempos dada a sua constante utilização e massificação.

Essa dependência obriga à criação de mecanismos que garantam uma elevada fiabilidade no seu funcionamento. No entanto, um sem número de problemas emergem constantemente tendo por base questões relacionadas quer com a própria estrutura, quer com a sua utilização indevida. Um dos problemas reside na própria vulnerabilidade das redes dos sistemas informáticos, sendo que os ataques se baseiam quase sempre no aproveitamento de defeitos dos sistemas e do *software* aplicacional que está instalado. Também existem problemas associados à má gestão na administração dos sistemas. A tudo isto acrescenta-se o facto de muitas das vezes ser negligenciada a segurança dos sistemas, não sendo adoptados os mecanismos de segurança adequados à sua protecção (Mendyk-Krajewska & Mazur, 2010)

Sendo assim, torna-se crucial que os responsáveis informáticos tomem consciência dos efeitos devastadores de um ataque ou ameaça aos seus sistemas de informação. (Magalhães & Grilo, 2006)

Assim, neste capítulo são abordadas as principais ameaças à segurança da informação, tais como a modificação, repetição, negação de serviço, intercepção e repúdio. São igualmente expostas as características que a informação deve deter de forma a evitar essas ameaças. No próximo subponto é abordado o tema das ameaças à segurança.

4.2 Ameaças à Segurança

Durante anos, as instituições que lidam com a problemática da segurança informática forneceram dados sobre o aumento dos ataques nos sistemas informáticos, sendo que esse aumento, cresceu nos últimos tempos de forma preocupante. Vários relatórios que foram publicados por diversos organismos demonstram que o rápido aumento destas ameaças incide sobre *spam*, código malicioso, técnicas de engenharia social e *phishing*. (Mendyk-Krajewska & Mazur, 2010)

Os autores (Magalhães & Grilo, 2006) definem a ameaça dentro do contexto informático, como sendo uma acção que tem o desígnio de comprometer a segurança do fluxo da informação que transita entre duas entidades. Segundo (Mendyk-Krajewska & Mazur, 2010), as ameaças que residem sobre a violação de direitos de autor e roubo de dados pessoais, de acordo com o Centro de Pesquisa Roubo de Identidade, tiveram o seu maior pico no ano de 2008.

Apresenta-se em seguida um exemplo de uma situação simples, ilustrada na Figura 4.1, no qual o emissor envia uma mensagem a um receptor com informação secreta. Se nesta situação se considerar que existe um terceiro participante, sendo neste caso o atacante, aquele que pretende realizar um ataque, este pode incidir sobre um destes pontos: (Magalhães & Grilo, 2006)

- A mensagem;
- O canal de comunicação;
- A sua infra-estrutura (emissor/receptor).

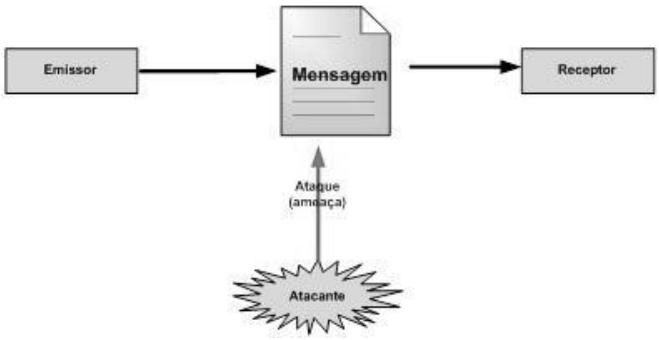

Figura 4.1 - Esquema de um ataque a uma mensagem (adaptada de (Magalhães & Grilo, 2006)

Caso não se consiga ter confiança no emissor ou no receptor, sendo o atacante um deles, pode haver uma entidade independente que assegura a segurança na comunicação e em que ambos os participantes confiam, conhecidas como *TTP (Trusted Third Party)*.

Assim, podem-se classificar em seis categorias os ataques à informação. São eles:

- Modificação;
- Repetição;
- Intercepção;
- Disfarce;
- Repúdio;
- Negação de serviço.

4.2.1 Modificação

Este tipo de ataque é baseado na alteração dos dados da mensagem em trânsito, como se pode verificar na Figura 4.2. A mensagem pode ser adulterada por acidente ou de forma maliciosa. (Vaithiyanathan, Gracelin, Edna, & Radha, 2010)

De forma a ilustrar este tipo de ataque, pode-se considerar a alteração do conteúdo da mensagem realizada por uma terceira entidade, que, embora não estando autorizada, altera o seu conteúdo. (Magalhães & Grilo, 2006)

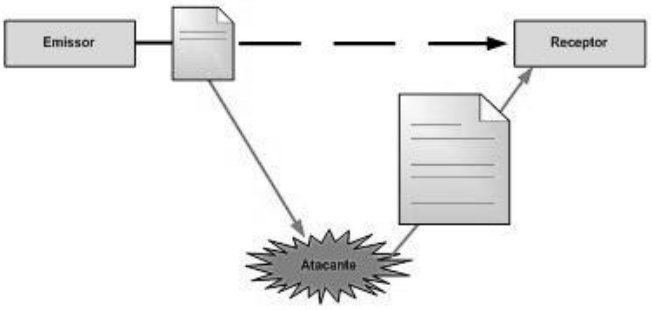

Figura 4.2 - Esquema de um ataque de modificação (adaptada de (Magalhães & Grilo, 2006))

4.2.2 Repetição

Este tipo de ataque ocorre quando uma operação já se encontra efectuada (concluída). Esta é novamente repetida, sem autorização do emissor, mas de forma a obter o mesmo resultado, como exemplificado na Figura 4.3.

Por exemplo, um fornecedor utiliza constantemente os dados enviados pelo comprador para fazer o pagamento da encomenda. Desta forma o atacante tenta obter de forma ilegal, gratificações adicionais. (Magalhães & Grilo, 2006)

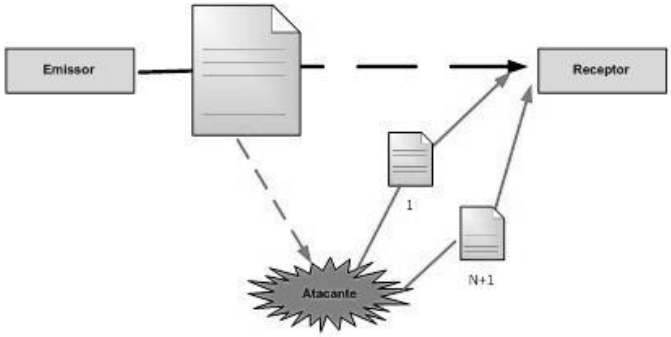

Figura 4.3 - Esquema de um ataque de repetição (adaptada de (Magalhães & Grilo, 2006))

4.2.3 Intercepção

Este ataque surge com o acesso não autorizado a uma mensagem, embora não existindo a possibilidade de a alterar (**Erro! A origem da referência não foi encontrada.**), como, por xemplo, no caso de uma troca de informação confidencial entre dois colaboradores de uma determinada empresa que são "espiados" por uma empresa concorrente que visa a obtenção dessa informação. (Magalhães & Grilo, 2006)

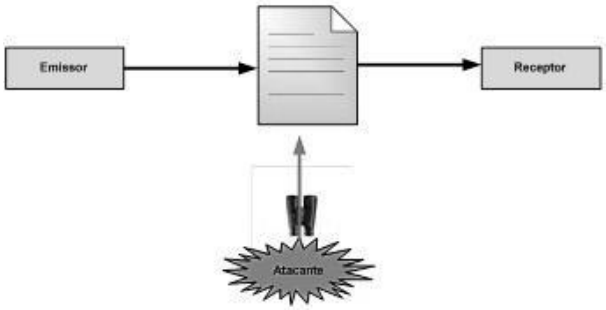

Figura 4.4 - Esquema de um ataque de intercepção (adaptada de (Magalhães & Grilo, 2006))

Aaaa

4.2.4 Disfarce

Este ataque é realizado com a apresentação de uma falsa identidade para um determinado atacante a um receptor, como exemplificado na Figura 4.5. Esta situação pode ocorrer quando o atacante quer ocultar a sua própria identidade ou assumir a identidade do emissor, com o intuito de lesar o emissor. (Magalhães & Grilo, 2006)

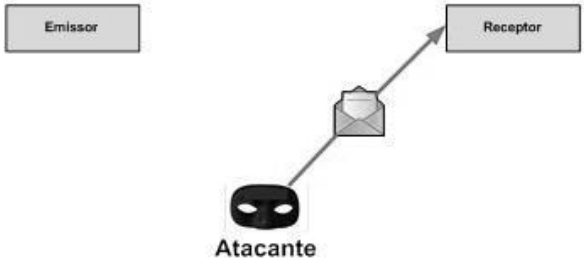

Figura 4.5 - Esquema de um ataque de disfarce (adaptada de (Magalhães & Grilo, 2006))

4.2.5 Repúdio

Baseia-se na negação de fazer parte duma comunicação ou transacção, quando na realidade se é parte activa (Figura 4.6). Esta situação verifica-se quando um qualquer cliente recusa a autoria e/ou envio de uma mensagem certificando o pagamento, ou, por sua vez, o vendedor nega ter recebido a anulação de uma encomenda. (Magalhães & Grilo, 2006)

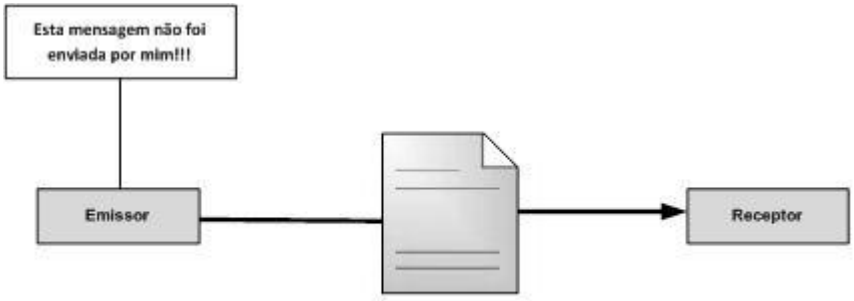

Figura 4.6 - Esquema de um ataque de repúdio (adaptada de (Magalhães & Grilo, 2006))

4.2.6 Negação de serviço

Este ataque tem como objectivo alterar o bom funcionamento de um sistema, sobrecarregando uma infra-estrutura de comunicações, tornando assim a sua utilização quase impossível. Porém existe outro tipo de ataque cujo objectivo visa bloquear um determinado endereço específico (Figura 4.7). (Magalhães & Grilo, 2006)

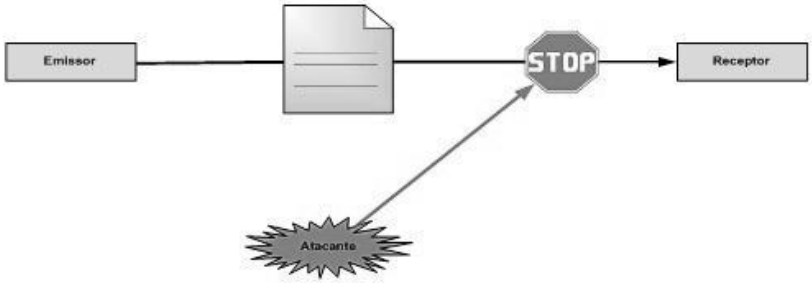

Figura 4.7 - Esquema de um ataque de negação (adaptada de (Magalhães & Grilo, 2006))

No caso dos ataques de Negação de Serviço, a ameaça pode surgir num nó ou em vários, bastando que estes estejam sob controlo de um atacante. Desta forma, o tráfego dos dados de destino pode não ser capaz de usar os serviços, pois estes são afectados pelo ataque de negação de serviço. Um ataque deste tipo pode afectar de forma selectiva, aleatória ou genérica os fluxos de comunicações. (Magalhães & Grilo, 2006)

Isto provoca que os pedidos que surgem de fontes seguras, não obtenham a melhor resposta por parte do sistema, como teriam caso não existisse o ataque. A solução passará pelo bloqueio/negação de acesso a determinadas portas ou endereços IP através, por exemplo, de uma *firewall*. (Srinivas & Chan, 2003)

De forma a organizar as seis categorias de ataque, estas podem ser agrupadas em duas classes, conforme ilustrado na Figura 4.8: (Magalhães & Grilo, 2006)

- Ataques activos;
- Ataques passivos.

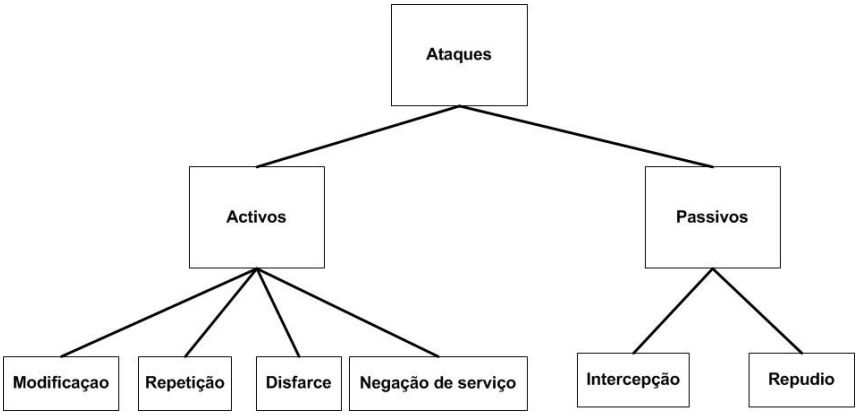

Figura 4.8 - Classificação dos ataques de acordo com o método (Magalhães & Grilo, 2006)

4.3 Mecanismos de segurança

Neste ponto, e quando na generalidade se considerarem os detalhes que a informação deve ter para garantir a sua segurança, estes mecanismos podem ser classificados sob os títulos: confidencialidade, integridade, autenticação, autorização, registo e não repúdio.

4.3.1 Confidencialidade

Esta propriedade consiste na protecção de informação sensível ou privada contra um ataque de intercepção, ou seja, contra acessos não autorizados. Em geral, essas garantias obtêm-se através da codificação dos dados utilizando algoritmos de cifra. (Vaithiyanathan et al., 2010)

4.3.2 Integridade

A integridade constitui a protecção da informação contra um ataque de modificação. (Vaithiyanathan et al., 2010) Numa comunicação entre dois interlocutores, pode conseguir-se garantir segurança, ou pelo menos, detectar que ocorreu uma modificação, recorrendo à utilização de algoritmos de sumário. (Magalhães & Grilo, 2006)

4.3.3 Autenticação

Compreende a protecção contra o disfarce da identidade de um dos intervenientes para que numa comunicação haja a garantia dos intervenientes serem quem dizem ser. (Vaithiyanathan et al., 2010)

A autenticação pode ser obtida através da utilização de: (Magalhães & Grilo, 2006)

- Sigilos entre os intervenientes, exemplo de senhas ou combinações de nome de utilizador e palavra-chave;
- Aparelhos únicos como *Tokens* de segurança, *smartcards* e cartões-matriz.

Um *Token* de segurança, que por vezes é designado por *Token* de *hardware*, *Token* de autenticação ou *Token* criptográfico, é um aparelho (Figura 4.9) que um utilizador usa por forma a ter permissão de acesso a um serviço, por exemplo, numa *VPN (Virtual Private Network)*. A forma de validação pode ser simples, usando apenas o número fornecido pelo equipamento, ou pode conjugar-se com o que se designa de "*two factor*", que consiste num *PIN (Personal Identification Number)* pessoal (parte privada) e do número que o *Token* apresenta no visor (parte publica). (de Borde, 2008)

Figura 4.9 - Exemplo de Token (Fonte: http://www.rsa.com/products/securid/datasheets/2305_h9061-sid-ds-0212.pdf)

Um *smartcard* (Figura 4.10) é um pequeno cartão de plástico com um microprocessador incorporado de modo a ter capacidade de armazenamento, memória e capacidade de gestão e execução de um algoritmo de criptografia. Um *smartcard* oferece segurança e portabilidade dos dados, sendo considerado mais seguro que um cartão de banda magnética. (Taherdoost et al., 2009)

Figura 4.10 - Exemplo de um smartcard (Fonte: http://www.andreonicards.com/contactcards.htm)

Um cartão matriz (Figura 4.11), é um pequeno cartão que contém uma matriz de elementos que permitem a um utilizador a realização de determinadas operações, como, por exemplo, transacções bancárias realizadas *online*.

Figura 4.11 - Exemplo de um cartão tipo matriz (Fonte: http://www.montepio.pt/SitePublico/pt_PT/particulares/solucoes/montepio24/seguranca/cartao-matriz.page?altcode=10006P)

4.3.4 Autorização

Autorização refere-se à característica que garante a protecção da informação contra acções não autorizadas. Esta garante, por exemplo, que a informação só seja acedida apenas por um número restrito de utilizadores, e que estes não podem executar acções para as quais não estão habilitados, garantindo que quem realiza as operações está devidamente credenciado. (Magalhães & Grilo, 2006)

4.3.5 Registo

No mecanismo de registo é permitido o arquivo de certas operações para observação à posteriori, de modo a obter um registo para saber quem fez o quê, e analisar quando se detecta algum problema na actividade de um certo serviço ou sistema. Por exemplo, na utilização de um cartão de crédito em negócios electrónicos, é importante que existam registos de todas as transacções, para que se alguém não autorizado fizer uso impróprio de um cartão, a acção seja facilmente descoberta. (Magalhães & Grilo, 2006)

4.3.6 Não repudio

É a característica que consiste na protecção contra a negação da participação numa determinada operação. (Vaithiyanathan et al., 2010) O acto de não repúdio pode ser realizado em três fases distintas, nomeadamente: (Magalhães & Grilo, 2006)

1. Na criação, quando o autor não possa negar a sua criação nem o seu envio, por exemplo, caso o documento enviado esteja assinado;

2. Na submissão, quando o autor consegue uma prova do seu envio como, por exemplo, no correio registado;

3. Na recepção, quando o destinatário não pode contestar que não a recebeu, por exemplo, o correio registado com aviso de recepção.

4.4 Conclusão

Neste capítulo sobre segurança informática foi realizada uma introdução ao tema, abordou-se a temática das ameaças à segurança, onde foram desenvolvidas questões como a repetição, intercepção, disfarce, repúdio e negação de serviço. Nos mecanismos de segurança foram abordadas a autenticação, a autorização, o registo e, por fim, o não repúdio. O próximo capítulo é dedicado à criptografia.

5 Criptografia

Neste capítulo serão abordados aspectos relacionados com o tema da criptografia, sendo que numa fase inicial é efectuada uma introdução ao tema, seguida de conceitos e noções básicas. De seguida é apresentado o tema de algoritmos de cifra, onde surge os algoritmos de chave simétrica. Neste ponto é abordado o tema da encriptação por fluxo e blocos.

Subsequentemente são abordados o tema dos algoritmos de chave assimétrica, com especial ênfase no *RSA (Rivest, Shamir, & Adleman)*, e as temáticas de algoritmos de sumário, assinaturas digitais, certificados digitais e, por último, as entidades certificadoras.

5.1 Noções básicas de criptografia

A definição do tema criptografia, palavra que advêm do grego *kryptós*, que significa escondido, e *gráphien*, que significa escrever, é a ciência que se destina ao estudo dos métodos e algoritmos pelos quais texto legível é modificado num qualquer ilegível. Desta forma garante-se que o texto lido pelo receptor é o desejado, e que só quem conhece o seu segredo (chave), tem acesso ao mesmo. (Magalhães & Grilo, 2006)

Nos sistemas criptográficos actuais, a criptografia abrange muito mais do que apenas codificar e descodificar. Com a massificação da utilização da *Internet* para realizar negócios (vender, comprar), tornou-se necessário garantir a privacidade nas trocas de correio-electrónico, nos pagamentos com cartões de crédito, na consulta de páginas ou noutras operações que se pretenda realizar, de modo a que estas não estejam vulneráveis a qualquer tipo de acções abusivas de pessoas ou organizações, com os conhecimentos adequados. (Magalhães & Grilo, 2006)

A criptografia cria para isso esquemas e protocolos que garantem que certas tarefas podem ser executadas mesmo na presença de possíveis atacantes. Assim, uma tarefa básica da criptografia é garantir que possa existir uma comunicação segura sobre um meio de transmissão inseguro, e que esta se proceda com garantia de autenticação e privacidade. (Coron, 2006)

De forma a garantir-se uma utilização correcta e segura da criptografia, tem que se garantir sob que forma o atacante pode ou não intervir sobre uma conversação. Isto é, saber o que ele pode ou não fazer, como, por exemplo, alterar ou usar informação a seu favor. Assim pode-se afirmar que um ataque pode ser bem ou mal sucedido através de uma fórmula matemática, com a excepção de alguma negligência por parte de quem garante a segurança. (Coron, 2006)

Para (Magalhães & Grilo, 2006), uma cifra é um algoritmo criptográfico, por exemplo, uma função matemática injectiva que faz alterações entre o texto original e o texto codificado

(cifrado) e vice-versa. Assim, não se utiliza uma função, mas sim um conjunto de funções indexadas por um parâmetro chamado chave. No caso da criptografia clássica, a não revelação dos pormenores da cifra garantia a sua segurança. Nos dias de hoje, o algoritmo é do conhecimento público e a capacidade da sua cifra avalia-se pelo tempo que esta demora a ser quebrada em ataques.

Os ataques mais clássicos são o da força bruta e o da análise de frequências. No primeiro, o método utilizado é o de testar todas as combinações exequíveis de símbolos até que se encontre a chave que vai permitir a descodificação do texto. No segundo, este vai basear-se na procura de certos caracteres, ou combinações deles, que ocorrem mais frequentemente em certas linguagens. (Magalhães & Grilo, 2006) No próximo ponto será abordado o tema dos algoritmos de cifra.

5.2 Algoritmos de cifra

Os algoritmos de cifra são muito importantes para a criptografia, pois estes são usados como já referido para garantir segurança e privacidade nas comunicações. (Sakalli et al., 2004)

Um algoritmo de cifra é um conjunto de procedimentos matemáticos, em que as técnicas criptográficas se fundamentam. A chave de um algoritmo fornece a informação necessária para aplicar esses procedimentos de forma única. Os tipos de chaves utilizados são: (Magalhães & Grilo, 2006)

- Secretas:
- Públicas.

5.2.1 Algoritmo de chave Simétrica

Nos algoritmos de chave simétrica utilizamos a mesma chave (secreta) para cifrar e decifrar uma mensagem, como podemos ver na Figura 5.1, e segundo os autores (Magalhães & Grilo, 2006), a comunicação utilizada nestes algoritmos é processada da seguinte forma:

- O emissor e receptor optam em segredo por uma cifra e uma chave;
- O emissor cifra a mensagem e remete-a para o receptor;
- O receptor recebe a mensagem e procede à sua decifração.

Para (de Canniere et al., 2006) a encriptação simétrica tem como objectivo garantir que a mensagem que circula num canal de comunicação inseguro tenha sigilo, e normalmente esta ocorre em duas transformações matemáticas que são a encriptação e desencriptação. Assim, numa comunicação em que dois intervenientes possuem uma chave secreta, esta serve para

cifrar no primeiro caso e, depois, no receptor, para decifrar. Garante-se, desta forma, que qualquer atacante não compromete a segurança da mensagem enviada.

Na criptografia clássica, este tipo de algoritmos eram usados na protecção de mensagens de cariz militar, financeiro, politico, entre outros, usando técnicas que possibilitam a transformação de um texto original num texto codificado, técnicas tais como a:

- Substituição;
- Transposição.

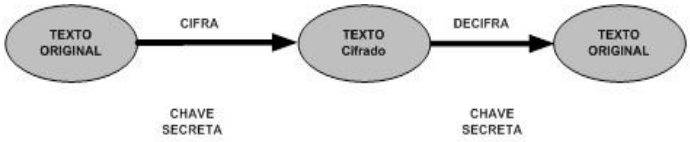

Figura 5.1 - Esquema de um algoritmo de chave simétrica (adaptada de (Magalhães & Grilo, 2006))

Em pleno século *XXI*, a comunicação entre o emissor e o receptor deixou de ser realizada em papel, na maioria dos casos, utilizando-se os computadores, nos quais a informação é guardada em sequência binárias. Um exemplo é o código *ASCII (American Standard Code for Information Interchange)*, que reproduz cada caracter através de oito dígitos binários (oito bits). (Magalhães & Grilo, 2006)

Assim sendo, os algoritmos de chave simétrica podem ser divididos da seguinte forma:

- Encriptação por blocos;
- Encriptação de fluxo, ou sequencial.

A encriptação por blocos reside em juntar os *bits* de uma mensagem em blocos de tamanho fixo, por exemplo, blocos de 64 *bits* ou 128 *bits*, codificando-os de forma independente. (de Canniere et al., 2006)

Podem ocorrer casos em que o tamanho do texto original não seja múltiplo do tamanho pré-fixado, sendo o último bloco completado de acordo com uma regra predeterminada conhecida por *padding* (ver Figura 5.2). (Magalhães & Grilo, 2006)

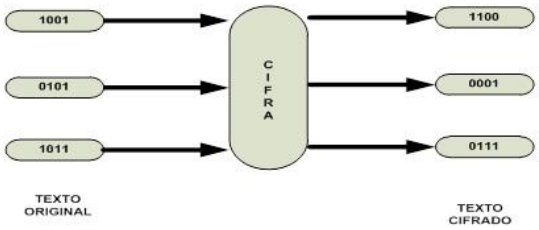

Figura 5.2 - Esquema de um algoritmo de encriptação por blocos (adaptada de (Magalhães & Grilo, 2006))

No que diz respeito à encriptação de fluxo, esta assenta no processamento contínuo de um *bit* ou um *byte* de uma mensagem de cada vez, (de Canniere et al., 2006) combinando-a com uma sequência de chaves que é gerada aleatoriamente, como exemplificado na **Erro! A origem da eferência não foi encontrada.**. Pode-se assim dizer que a chave da cifra funciona como estado inicial do gerador de chaves. (Magalhães & Grilo, 2006)

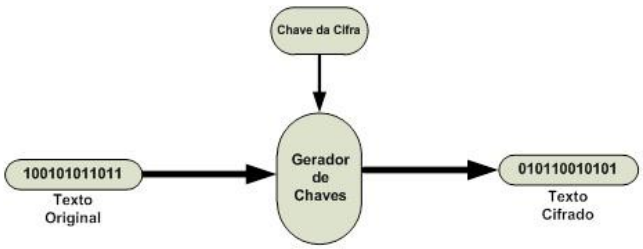

Figura 5.3 - Esquema de um algoritmo de encriptação de fluxo (adaptada de (Magalhães & Grilo, 2006))

A encriptação de fluxo é efectuada com maior velocidade que a encriptação por blocos e precisa de menos *hardware.* No entanto, a encriptação de fluxo pode levar a graves problemas de segurança, como, por exemplo, quando o estado inicial da chave é repetido em diversas comunicações. Apesar disso, a encriptação de fluxo deverá ser escolhida nas situações em que se desconheça o tamanho do texto. Isto não quer dizer que não se possa também utilizar a encriptação por blocos, dependendo a escolha das transmissões eficientes e complexidades acrescidas que se pretenda. (Magalhães & Grilo, 2006)

A encriptação de fluxo é efectuada com maior velocidade que a encriptação por blocos e precisa de menos *hardware*. No entanto, a encriptação de fluxo pode levar a graves problemas de segurança, como, por exemplo, quando o estado inicial da chave é repetido em diversas comunicações. Apesar disso, a encriptação de fluxo deverá ser escolhida nas situações em que se desconheça o tamanho do texto. Isto não quer dizer que não se possa também utilizar a encriptação por blocos, dependendo a escolha das transmissões eficientes e complexidades acrescidas que se pretenda. (Magalhães & Grilo, 2006)

Uma das vantagens das implementações baseadas em algoritmos simétricos é que estes são mais eficientes do que os algoritmos assimétricos (chave publica). (de Canniere et al., 2006)

Um problema associado aos algoritmos de chave simétrica é a gestão eficiente das chaves, uma vez que elas têm de se conservar secretas antes, durante e depois de uma comunicação. Uma solução possível consiste em gerar e distribuir antecipadamente as chaves secretas necessárias. (Magalhães & Grilo, 2006)

Segundo (Magalhães & Grilo, 2006), para uma comunicação com **N** elementos será indispensável produzir, proteger e armazenar $(\frac{N(N-1)}{2})$chaves, sendo imprescindível uma chave por cada par diferente de utilizadores. Para se conseguir uma optimização dos recursos basta dar a um dos elementos o encargo do armazenamento de todas as chaves existentes e da sua comunicação aos outros elementos, sempre que tal for indispensável. Alternativamente, pode-se usar uma chave de sessão, que é produzida em cada comunicação entre o emissor e receptor, e que é eliminada logo após o fim da sessão. (Magalhães & Grilo, 2006)

O protocolo de Diffie-Hellman permite fazer a troca de chaves secretas, que podem ser ou não de sessão, através de canais públicos, como é descrito nos passos seguintes: (Magalhães & Grilo, 2006)

- O emissor e o receptor elegem dois números primos, de grandes dimensões, p e $g < p$;
- O emissor gera aleatoriamente um número $k_e < p$ e envia $E = g^{k_e}(mod\ p)$ para o receptor;
- O receptor gera aleatoriamente um número $k_r < p$ e envia $R = g^{k_r}(mod\ p)$para o emissor;
- Ambos calculam $K = E^{k_r}(mod\ p) = R^{k_e}\ (mod\ p)$ que representa a chave.

Refira-se que, mesmo conhecidos os valores de *p, g, E* e *R*, disseminados através de canais públicos, não é possível achar *K* sem conhecer k_e e/ou k_r. Este protocolo abre caminho aos algoritmos de chave assimétrica que irão ser descritos no próximo ponto. (Magalhães & Grilo, 2006)

Para finalizar este ponto, aborda-se o método de encriptação *DES/3DES*. Originalmente o *DES (Data Encryption Standard)* foi desenvolvido nos anos 70 pela *IBM* e ainda hoje é usado em sistemas *UNIX* e em sistemas de caixas multibanco. O *DES* utiliza uma chave simétrica de 56 *bits* e é usada em blocos de dados com 64 *bits*. O processo é rápido mas considerado inseguro por alguns especialistas. O *3DES (Triple Data Encryption Standard)* usa o padrão de encriptação do *DES*, mas recorrendo um método que permite uma cifragem tripla, produzindo uma chave com um tamanho de 112 a 168 *bits*. (Mostafa et al., 2008)

Outro método de encriptação é o *AES (Advanced Encryption Standard)* que utiliza encriptação por blocos e foi adoptado como padrão pelo governo dos Estados Unidos da América em finais do ano 2001, e que usa uma chave de tamanho variável de 128, 192 ou 256 *bits*, aplicada em blocos de 128 *bits*. É o sucessor do *DES*. (Magalhães & Grilo, 2006)

5.2.2 Algoritmos de chave Assimétrica

Estes algoritmos utilizam duas chaves complementares, uma pública e uma outra privada, de forma a cifrar e a decifrar o texto, como se pode ver na **Erro! A origem da referência não foi ncontrada.**. (Magalhães & Grilo, 2006)

Figura 5.4 - Esquema de um algoritmo de chave assimétrica (adaptada de (Magalhães & Grilo, 2006))

A comunicação destes algoritmos segue os seguintes passos: (Magalhães & Grilo, 2006)

- O emissor e o receptor elegem uma cifra;
- O receptor envia, por rede pública, a sua chave pública;
- O emissor cifra o texto com a chave pública do receptor e envia-o;
- O receptor decifra o texto com a sua chave privada.

As funções *one-way* (de sentido único) são a base da criptografia de chave pública. São funções muito simples de calcular mas impossíveis de inverter. Sendo assim, a utilização destas funções em qualquer texto no processo de encriptação é muito fácil, mas o processo de desencriptação é difícil, sendo este possível somente no receptor. São utilizadas subclasses de funções *one-way*, que são denominadas por *one-way* com *trapdoor*, que admitem a inversão, desde que se conheça o segredo. Todo o emissor pode cifrar o texto desde que conheça a função, mas só os receptores que possuem o segredo (chave) podem efectuar a desencriptação. No caso dos algoritmos de chave simétrica, estes são como um cofre, sendo que apenas dois utilizadores têm a chave. O emissor põe o texto no interior do cofre, sendo esta retirada pelo receptor. (Magalhães & Grilo, 2006)

Os algoritmos de chave assimétrica são assim uma espécie de caixa de correio, existindo a possibilidade de qualquer pessoa enviar uma mensagem, bastando para tal conhecer o endereço (chave pública), mas apenas o proprietário da caixa de correio possui a chave (chave privada) que autoriza a abertura das mensagens. (Magalhães & Grilo, 2006)

O *RSA* é um exemplo de algoritmo de chave assimétrica muito utilizado em protocolos de negócio electrónico. Foi desenvolvido nos Estados Unidos da América na década de 70. As patentes deste algoritmo já são do domínio público, sendo usado como parte do *software OpenSSL*. Apesar das várias tentativas de ataque contra chaves de comprimento de 128 *bits*, estas não foram bem-sucedidas. No entanto, e mesmo demonstrando-se ser seguro, com o aumento da capacidade computacional as chaves aumentaram para 1024 ou 2048 *bits*. (Romney & Parry, 2006)

A segurança do algoritmo *RSA* está na complexidade de factorizar em termos computacionais um número primo de grande dimensão. Desta forma, o algoritmo que gera as chaves é bastante simples consistindo em: (Menezes et al., 1996)

- Escolha de dois números primos, p e q, em que $p \neq q$;
- Calcular $n = pq$;
- Escolher aleatoriamente um número que seja primo com $(p-1)(q-1)$.

Após estes passos terem sido concluídos, as chaves públicas e privadas são formadas pelos subsequentes pares:

- Chave pública: (n, e);
- Chave privada: (n, d) Obtêm-se $d = e^{-1}(mod\ (p-1)(q-1)$.

Os processos de encriptação e desencriptação de uma mensagem m:

- Encriptação: $c = m^e mod(n)$;
- Desencriptação: $m = c^d mod\ (n)$.

Usando os seguintes valores segundo (Menezes et al., 1996):

- p=47;
- q=71;
- n=pq=3337;
- e=79;

- *d*=1019.

A encriptação do texto original $\mathrm{m} = 688$ corresponde a $\mathrm{c} = 688^{79} \bmod 3337 = 1570$. No caso de se usar a função inversa, então $m = 1570^{1019} mod(3337)$, e no final obtêm-se o texto original.

5.2.3 Algoritmo de sumário

Os algoritmos de sumário são geralmente chamados de funções *Hash*. Têm um papel basilar na criptografia moderna, por exemplo, no fabrico de assinaturas digitais. O objectivo das funções *Hash* é modificar de forma susceptível a mensagem original, que exibe um tamanho variável, no sumário de tamanho fixo. (Yang et al., 2009)

Uma característica das funções *Hash* é serem não invertíveis, embora seja inexequível obter o texto original a partir do sumário. No entanto, é possível que duas mensagens diferentes proporcionem o mesmo sumário, pois as funções não são injectivas. Os algoritmos de cifra têm como principal objectivo assegurar a confidencialidade da mensagem. Estes apenas tentam garantir a integridade. (Magalhães & Grilo, 2006)

A comunicação utilizada nestes algoritmos é descrita nos seguintes passos: (Magalhães & Grilo, 2006)

- O emissor e receptor elegem uma função de *Hash*;
- O emissor emite a mensagem juntamente com o sumário;
- O receptor determina o seu próprio sumário e confronta-o com o original. Caso não sejam idênticos, prova-se que a mensagem foi alterada na circulação.

No parágrafo seguinte são apresentados alguns exemplos de algoritmos de sumário, começando pelo *MD2 (Message-Digest Algorithm 2)*, desenvolvido em 1989, o *MD4 (Message-Digest Algorithm 4)* em 1990 e, no ano seguinte, o *MD5 (Message-Digest Algorithm 5)*. Tendo sido detectada uma debilidade no *MD5* em 1996, os especialistas recomendaram a utilização de outros algoritmos, como o *SHA-1 (Secure Hash Algoritm)* e o *RIPEMD (Race Integrity Primitives Evaluation Message Digest)*.

O grupo de algoritmos *MD (Message-Digest Algorithm)* gera sumários de 128 *bits* e têm sido bastante usados para garantir a integridade dos ficheiros descarregados através da Internet. O grupo de algoritmos *SHA* tem no *SHA-1* o algoritmo com maior visibilidade pública. Este produz sumários de 128 *bits*, sendo usado num grande leque de aplicações de segurança e está presente em protocolos (*SSL (Secure Sockets Layer)*, *TLS (Transport Layer Security)*, *PGP (Pretty Good Privacy)*, entre outros) Após terem sido realizados ataques ao *SHA-1*, foram desenvolvidos quatro algoritmos adicionais que são o *SHA-224*, *SHA-256*, *SHA-384*, *SHA-512*, que geram sumários de 224, 256, 384 e 512 *bits*, e são designados por *SHA-2*. Por último, o

RIPEMD é um algoritmo de sumário que tem por base o *MD4*, mas produz sumários de 160 *bits*. No entanto, existem versões deste algoritmo que produzem sumários de 128, 256 e 320 *bits*. (Magalhães & Grilo, 2006)

5.2.4 Assinatura digital

No contexto histórico e social, a assinatura manuscrita é usada como marca da autoria, em consonância com o teor de um documento. A sua constante falsificação motivou o surgimento da assinatura digital. Esta garante integridade, autenticidade e não-repúdio, e é aplicada nos documentos electrónicos. (Magalhães & Grilo, 2006)

Em seguida observam-se os procedimentos para utilização de uma assinatura digital (ver Figura 5.5): (Magalhães & Grilo, 2006)

- O emissor assina e gera o sumário;
- O emissor cifra o sumário com a sua chave privada e remete o documento juntamente com o sumário ao receptor;
- O receptor decifra o sumário usando a chave pública do emissor;
- O receptor apura a assinatura determinando o sumário a partir do documento original e comparando-o com o sumário decifrado.

Caso sejam iguais, assegura-se que a assinatura não é repudiável. Assim o emissor não pode refutar que enviou o documento, pois só ele sabe a sua chave privada. É de referir, que para a assinatura ser irrefutavelmente segura, é imprescindível fazer prova da sua veracidade. Para isso recorre-se a um sistema de certificação de chaves públicas. (Magalhães & Grilo, 2006)

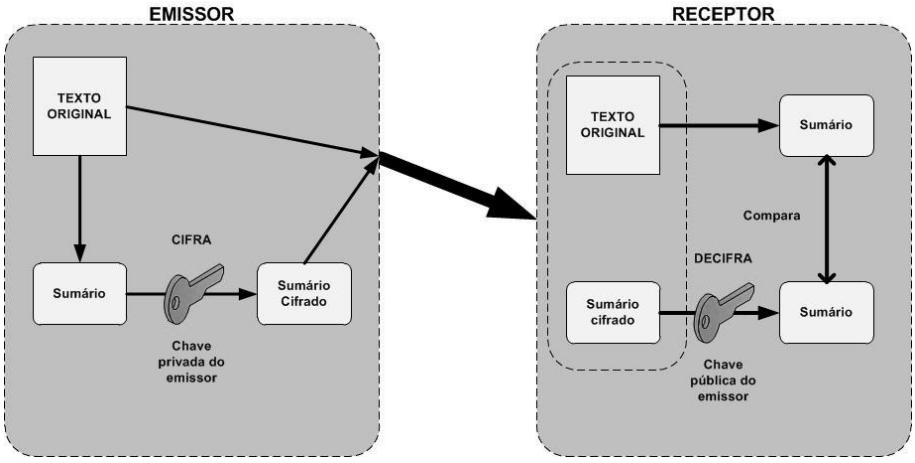

Figura 5.5 - Esquema de uma assinatura digital (Magalhães & Grilo, 2006)

Como exemplos de assinaturas digitais temos o *RSA* e o *DSA (Digital Signature Algorithm)*. No caso do *RSA* utiliza-se o algoritmo de chave assimétrica *RSA* com o algoritmo de sumário *MD5*.

Os cálculos exactos são os mesmos, mas neste caso a encriptação é realizada com chave privada e a desencriptação utiliza a chave pública. O *DSA* é o padrão de assinaturas digitais do governo Americano, é baseado no algoritmo de chave assimétrica *ElGamal* e utiliza o algoritmo de sumário *SHA-1*. Tem a desvantagem de ser mais lento que o *RSA*. (Magalhães & Grilo, 2006)

A utilização dos certificados digitais numa comunicação permite aos participantes poderem ter a certeza que cada vez que usam uma chave pública, a entidade com quem pretendem trocar informação possui uma chave privada associada. Um certificado digital é um conjunto de dados que reconhece uma entidade, seja ela uma pessoa, empresa ou um computador, e a respectiva chave pública. O certificado digital garante a autenticidade e não-repúdio na comunicação, certifica a veracidade dos dados contidos nele e é assinado digitalmente por uma entidade em que todos confiam (*TTP*). O formato mais usado nos certificados digitais é o *X.509* da *ITU (International Telecommunications Union)*, sendo composto por: (Magalhães & Grilo, 2006)

- Versão do certificado;
- Número de série;
- Algoritmo de assinatura;
- Emissor;
- Período de validade;
- Identificação do titular;
- Chave pública;
- Extensões.

Pode-se ver na Figura 5.6 um exemplo de um certificado digital.

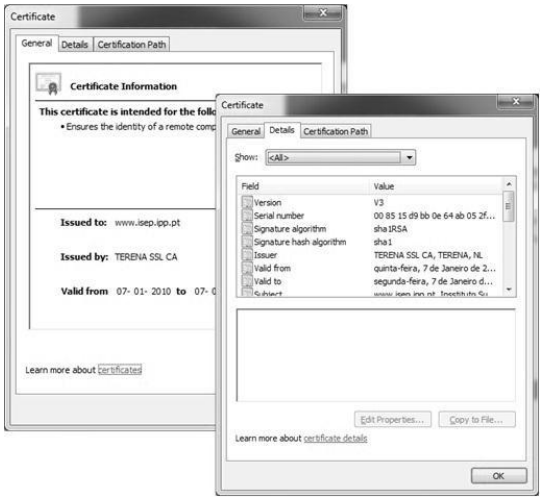

Figura 5.6 - Exemplo de um certificado digital *(ISEP.IPP.PT)*

Podem surgir situações em que é necessário proceder à revogação do certificado, por exemplo, caso a chave privada seja descoberta, ou algum dos dados do titular seja alterado. A informação sobre os certificados revogados pode ser obtida de duas maneiras: pelas *CRL (Certificate Revocation Lists)*; ou por uma consulta à *TTP* usando o *OCSP (Online Certificate Status Protocol)*. As *CRL* são normalmente de conhecimento de todos e de fácil acesso. A directoria *X.500* e servidores de *LDAP (Lightweight Directory Acess Protocol)* são exemplo desse tipo de repositórios. (Magalhães & Grilo, 2006)

O processo de verificação de uma assinatura digital tem os seguintes passos: (Magalhães & Grilo, 2006)

1. O receptor do documento adquire o certificado digital do assinante;
2. O receptor do documento consegue o certificado da *TTP*;
3. O receptor confirma a assinatura da *TTP* no certificado;
4. O receptor adquire a chave pública do assinante a partir do seu certificado;
5. O receptor autentica a assinatura digital dos dados.

No próximo ponto são abordadas as entidades certificadoras.

5.2.5 Entidades Certificadoras

As entidades certificadoras são entidades responsáveis pela emissão e confirmação dos dados presentes num certificado digital. Existem factores que podem aumentar a confiança numa entidade certificadora, particularmente: (Magalhães & Grilo, 2006)

- Processos de verificação dos dados (correio electrónico ou procuração);
- Entidades públicas como notários têm procedimentos uniformizados para realizar as verificações referidas no ponto anterior. São entidades que têm a confiança do público;
- Entidades privadas que respeitam as normas de verificação e de segurança, recorrendo à publicação de *CPS (Certification Practice Statements)*.

Por norma, a actividade destas entidades é subdividida em duas componentes:

- CA (Certification Authority);
- RA (Registration Authority).

A primeira é uma entidade que cria ou fornece condições para a geração e verificação de assinaturas digitais, fazendo a emissão e gestão do ciclo de vida dos certificados digitais, e ainda a sua publicação. No caso das *RA*, são entidades que reconhecem e certificam o detentor do certificado digital, promovem a celebração de contratos de emissão e gerem os certificados que não se encontrem atribuídos em exclusivo à *CA*. Por questões de segurança, em grande parte dos casos, a *CA* não está alcançável a partir do exterior, cabendo por isso à *RA* comunicar com ela. No caso dos navegadores *Web*, os certificados das *CA* são obtidos automaticamente. A Figura 5.7 apresenta alguns exemplos. (Magalhães & Grilo, 2006)

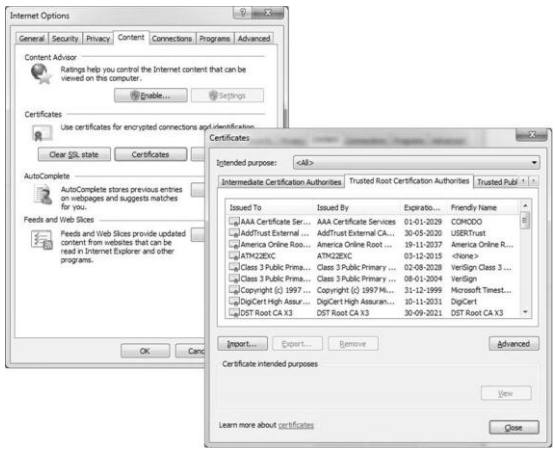

Figura 5.7 - *CA* pré-instaladas no navegador *Internet Explorer*

Para a utilização de um qualquer serviço que imponha o saber de uma chave pública, é indispensável a obtenção e validação do certificado digital. A validação afecta o conhecimento

da chave pública da *CA* que gerou o certificado. Existem, no entanto, alguns cenários onde um utilizador pode não validar o certificado directamente na *CA*, por exemplo, quando se obtém uma chave pública de forma insegura. Este tipo de situações pode ser solucionado caso o certificado esteja assinado por outra *CA* que seja conhecido pelo utilizador, formando-se uma cadeia de certificação em que uma *CA* certifica a veracidade do certificado de outra. Surge desta forma o conceito de *CA* raiz, funcionando como a raiz de confiança de todos os elementos que se encontram abaixo dela, como exemplificado na Figura 5.8. (Magalhães & Grilo, 2006)

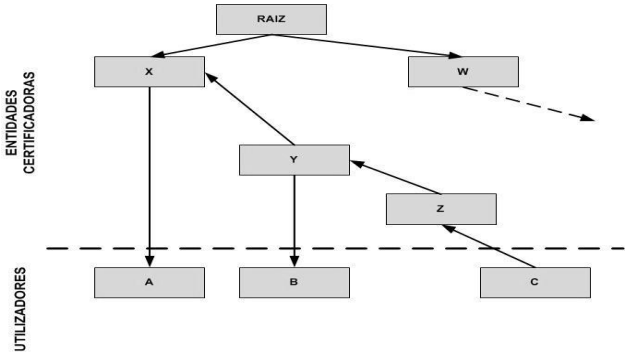

Figura 5.8 - Hierarquia de *certification authorities (CA)* (Magalhães & Grilo, 2006)

O *SET (Secure Electronic Transaction)* é um sistema que foi desenvolvido com o intuito de resolver o problema gerado pelo pagamento com cartões de crédito em redes abertas como a Internet. Como se pode observar na Figura 5.9, existe uma *CA* raiz na qual todos os participantes confiam. Assim, a autoridade certifica as marcas de cartões de crédito (*Visa*, *Mastercard* e *American Express*), que por sua vez certificam *CA* nas diferentes zonas do globo, chamadas de *CA* geopolíticas, que por seu lado vão certificar *CA* que geram certificados para os donos dos cartões de crédito (*cardholders*), comerciantes (*merchants*) e as entidades que consentem os pagamentos (*payment gateways*). (Magalhães & Grilo, 2006)

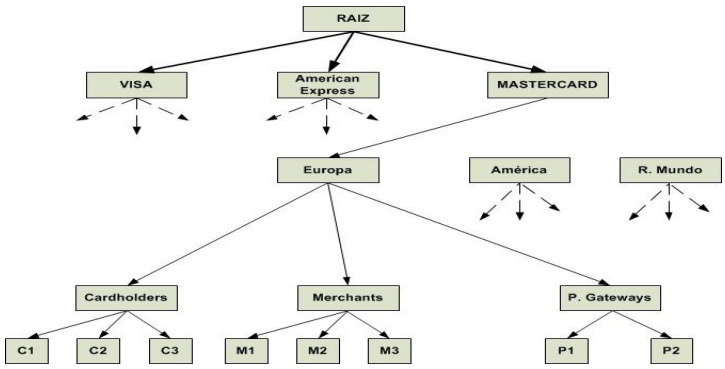

Figura 5.9 - Cadeia de certificação do *SET* (Magalhães & Grilo, 2006)

De forma a obter um ambiente confiável na comunicação numa rede de utilização pública, recorre-se às técnicas e conhecimentos relacionados com a criptografia assimétrica ou de chave pública. Para isso foram criados as infra-estruturas de chaves públicas (*PKI (Public Key Infrastructures)*). Estas infra-estruturas contêm *hardware*, *software*, utilizadores, políticas e

procedimentos necessários para gerir, criar, armazenar e revogar certificados de chave pública. Finalmente, e de forma a concluir este assunto, surge um modelo diferente de estabelecimento de certificados que não recorre a cadeias de certificação. Esse modelo é apelidado de rede de confiança e apoia-se nas relações pessoais entre os vários intervenientes. Pode-se dizer que é um modelo inseguro, pois as relações de confiança são informais e podem ser fraudulentas. Na Figura 5.10 é apresentado um exemplo do modelo de estabelecimento de redes de confiança. (Magalhães & Grilo, 2006)

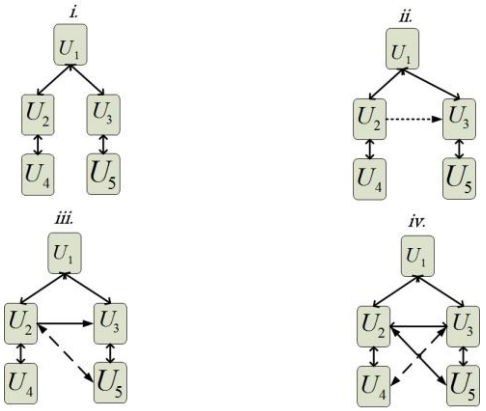

Figura 5.10 - Estabelecimento de redes de confiança (Magalhães & Grilo, 2006)

5.3 Conclusão

No capítulo da criptografia foram introduzidos temas relacionados com noções básicas, abordado o tema dos algoritmos de cifra, com destaque para os algoritmos de chave simétrica e assimétrica, estudadas as temáticas relacionadas com algoritmos de sumário, assinaturas digitais e por fim entidades certificadoras. No próximo capítulo é estudado o tema dos protocolos de rede seguros.

6 Protocolos de rede seguros

6.1 *Secure Sockets Layer* (SSL)

O nascimento do protocolo de comunicação *SSL* fica associado à criação da *Netscape*. Quando criou o seu primeiro navegador de *Web*, esta apercebeu-se da necessidade de garantir comunicações seguras entre os clientes e os servidores. Neste caso, a *Netscape* não conseguia introduzir uma encriptação directa no seu navegador *Web*. Este protocolo actua sobre o *TCP/IP*, como pode ser visualizado na Figura 6.1. (Chou, 2002)

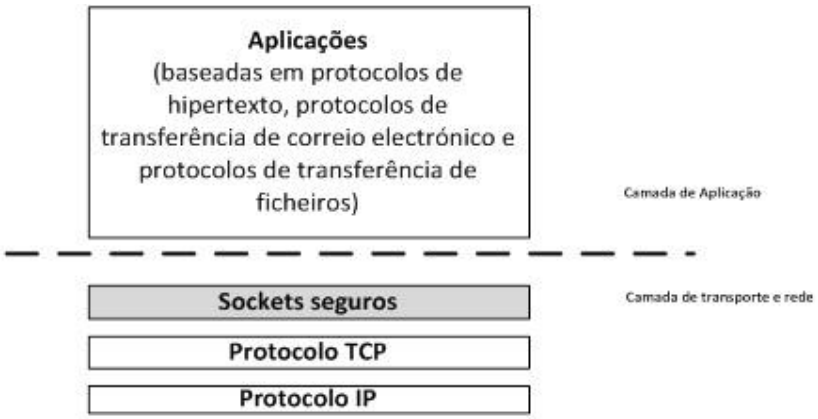

Figura 6.1 - Camada *SSL* (Chou, 2002)

Com a introdução deste protocolo de comunicação é possível garantir a integridade e confidencialidade nas comunicações realizadas na Internet. Este é usado para assegurar a segurança em ligações do tipo Telnet, *FTP (File Tranfer Protocol)* e *HTTP (HyperText Transfer Protocol)*. O seu modo funcionamento é baseado em sessões que se estabelecem entre um cliente e um servidor. O SSL é formado por dois subprotocolos: (Magalhães & Grilo, 2006)

- SSL handshake protocol;
- SSL record protocol.

No caso do primeiro, é usado pelos mecanismos de autenticação suportados no servidor perante o cliente, e vice-versa na versão 3.0. É utilizado também para transmitir certificados na norma *X.509* e estabelecer as chaves de encriptação dos dados. Na Figura 6.2 pode-se ver como funciona o *SSL handshake protocol*. (Magalhães & Grilo, 2006)

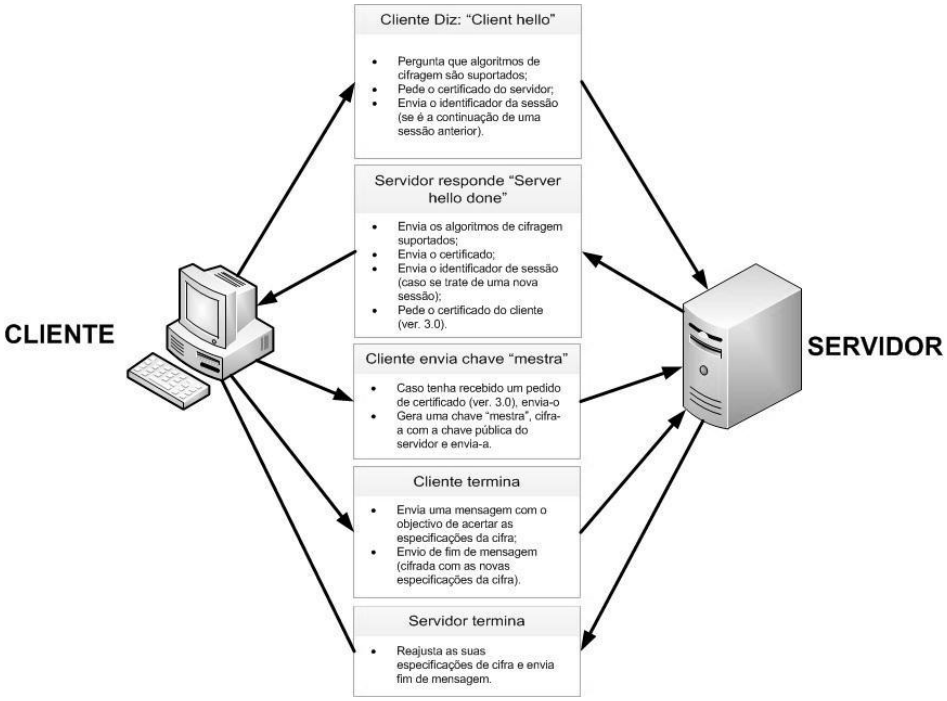

Figura 6.2 - *SSL handshake protocol* (Magalhães & Grilo, 2006)

No caso do *SSL record protocol*, este é usado no decorrer das sessões de transferência de dados que ocorrem entre o cliente e o servidor. É neste ponto que são descritos os formatos dos dados que serão transferidos e que são proporcionados mecanismos de compressão, encriptação e verificação da integridade. (Magalhães & Grilo, 2006)

Torna-se claro que o *SSL record protocol* é vital na segurança dos dados da aplicação. Todos os registos são compactados recorrendo a um algoritmo de compressão, estando este sempre activo. No entanto, o seu método é inicialmente definido como nulo *(null)*. No caso dos registos, estes são protegidos recorrendo a técnicas de criptografia e a algoritmos de *HMAC* que são descritos na especificação da cifra utilizada. (Kuihe & Xin, 2007)

Hoje em dias as assinaturas digitais e algoritmos que são suportados pelo *SSL* são: (Magalhães & Grilo, 2006)

- Algoritmos simétricos: *DES*, *3DES*, *AES*, entre outros;
- Algoritmos assimétricos: *RSA*;
- Algoritmos de sumário: *SHA-1* e *MD5*;
- Assinatura digital: *RSA* e *DSA*.

No presente o *SSL* tornou-se um padrão e é identificado pelos navegadores *Web* mais comuns através de um *URL (Uniform Resource Locator)* do género *HTTPS (HyperText Transfer Protocol Secure)* (ver Figura 6.3). Este processo é feito geralmente de forma automática pelo

navegador, tornando-o assim imperceptível para o cliente, garantindo a segurança na comunicação. (Kehe & He. Jianping, 2011)

Figura 6.3 - Exemplo de uma ligação *HTTPS*

No caso dos parâmetros criptográficos da sessão, estes são gerados pelo *SSL handshake protocol*. Quando se inicia uma ligação, na fase inicial é definida entre o cliente e o servidor qual a versão do protocolo que vão usar, existindo também opcionalmente a selecção dos algoritmos criptográficos que serão usados na autenticação entre ambos e as técnicas de chaves públicas para a geração de cifras partilhadas. O *SSL handshake protocol* é definido num nível superior do *SSL record protocol* e é usado para negociar os atributos de uma sessão segura. (Kuihe & Xin, 2007)

A Figura 6.4 reflecte a arquitectura da camada do protocolo *SSL*.

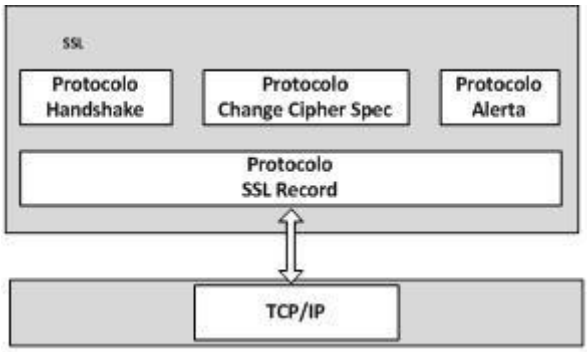

Figura 6.4 - Arquitectura da camada do protocolo *SSL* (Kuihe & Xin, 2007)

Mesmo com o *SSL handshake protocol* a implementar todos os mecanismos de segurança indispensáveis ao estabelecimento de uma ligação segura a um servidor, pode não ser suficiente para criar segurança na ligação. Considere-se como exemplo o pagamento de uma compra realizada na Internet, recorrendo ao pagamento através de um cartão de crédito, utilizando uma ligação *HTTPS*. (Magalhães & Grilo, 2006)

A ligação é segura, mas não se consegue garantir que o servidor o seja, pois o número do cartão de crédito introduzido pode ser armazenado numa base de dados, e esta pode ser furtada e usada pelo atacante em proveito próprio. Para estes casos devemos utilizar

componentes de segurança complementares, como, por exemplo, *firewalls*. (Magalhães & Grilo, 2006) No próximo ponto será abordado o tema *Transport Layer Security*.

6.2 Transport Layer Security (TLS)

O *TLS* é uma versão melhorada e actualizada do seu predecessor o *SSL*. Para o utilizador não existe qualquer diferença entre ambos, mas foram executadas melhorias internas em alguns algoritmos. Foi introduzido um mecanismo em que o *SSL* é usado caso o *TLS* não esteja disponível. (Magalhães & Grilo, 2006)

Relativamente ao seu funcionamento, o *TLS* pretende assegurar a segurança na comunicação numa rede pública. Neste processo é enviada por parte do servidor uma chave pública ao *browser*, para que este possa enviar uma chave secreta, que é criada aleatoriamente, ficando estabelecida a troca de dados entre os dois. (Hussain & Seret, 2006)

É baseado no conceito introduzido por *Diffie-Hellman* (chave pública) e usado em aplicações de email, navegação *Web*, conversa interactiva e *VOIP (Voice over Internet Protocol)*. Os algoritmos e as assinaturas digitais suportados pelo *TLS* são os mesmos do *SSL* com excepção da assinatura digital *DSA*. (Yasinsac & Childs, 2001)

A grande vantagem do *TLS* em relação ao seu predecessor (*SSL*) reside no facto de o *TLS* ser completamente livre de patentes. O *TLS* é composto pelo *Handshake protocol*, *Cipher Change protocol*, *Alert protocol* e o *Record protocol*, como se observa na Figura 6.5. (Portmann & Seneviratne, 2001)

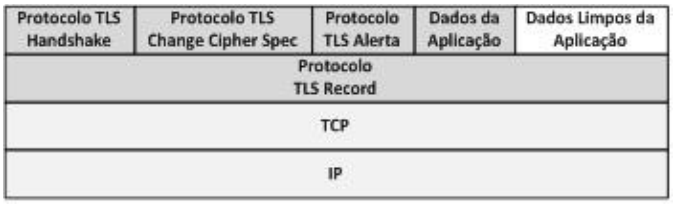

Figura 6.5 - Pilha Protocolar do TLS (Portmann & Seneviratne, 2001)

O *Record protocol* na sua camada garante confidencialidade através da chave simétrica, autenticidade e integridade nas mensagens e é o responsável pela fragmentação e eventual compressão das mensagens. (Hussain & Seret, 2006) Segundo os autores (Badra & Urien, 2008), este protocolo proporciona uma ligação básica para camadas superiores através do encapsulamento.

O *Handshake protocol* permite a autenticação do cliente e do servidor e negoceia os algoritmos a usar em cada ligação, possibilitando a troca de chaves simétricas para confidencialidade, autenticidade e integridade. O *Alert protocol* define o tipo de alerta (*warning* ou fatal) e faz a descrição do alerta. O *Cipher Change protocol*, quando recebido, aplica o *CipherSuite* negociado à ligação actual. (Portmann & Seneviratne, 2001)

6.3 Introdução ao IPSec

O *IPSec* é um padrão aberto desenvolvido pelo *IETF (Internet Engineering Task Force)* que pode ser usado com recurso a técnicas e serviços de criptografia. O *IPSec* é um protocolo da camada três no modelo *OSI*, que suporta autenticação, integridade, confidencialidade e protecção contra ataques de repetição. (Narayan et al., 2008)

Para compreender melhor o *IPSec* é obrigatório falar-se no *IP*. Este refere-se ao protocolo de transmissão de mensagens usado na *Internet*. Com ele define-se uma norma para a formatação de um cabeçalho. Estes são anexados aos dados que se pretendem transmitir, formando assim o que se chama de pacote. O conjunto de elementos relevantes num cabeçalho são os endereços de origem e destino dos pacotes, porque uma das funções do *IP* é encaminhar os pacotes desde a sua origem até ao seu destino. Porém, existem ataques à segurança que violam este princípio de funcionamento. A técnica é conhecida por *IP spoofing*, e consiste na modificação dos cabeçalhos, alterando-se os endereços de origem e destino. Assim sendo, de forma a solucionar este problema, foram definidos por parte do grupo de trabalho *IPSec* dois mecanismos de segurança: (Magalhães & Grilo, 2006)

- Autentication header;
- Encapsulating security payload.

O *Autentication header* fornece mecanismos de autenticação, integridade e contém o *SPI (Security Parameter Index)* dos pacotes. Por outro lado o *Encapsulating security payload* providencia a confidencialidade dos pacotes, cifrando o seu conteúdo, recorrendo à encriptação por blocos, sendo igualmente responsável pela segurança máxima quando usado em modo de túnel, assegurando a sua flexibilização. (Berger, 2006)

A ideia que está presente nos mecanismos de autenticação e confidencialidade para o protocolo *IP* é a associação de segurança, que mais não é que uma relação de sentido único entre o emissor e o receptor que relata quais os mecanismos de segurança (algoritmos de encriptação) que utiliza para criar uma ligação segura. Estes algoritmos baseiam-se no *IKE (Internet Key Exchange)*, o qual assenta num protocolo de *Diffie-Hellman* para a troca de chaves secretas, combinado com um certificado de chave pública para autenticação. (Magalhães & Grilo, 2006)

6.3.1 Datagramas IP

Os datagramas *IP* (pacotes) são as unidades fundamentais para a comunicação nas redes *IP*, sendo definidos dois cabeçalhos para o *IPSec*: o de encapsulamento de segurança *(ESP (Encapsulating Security Payload))*; e o de autenticação *(AH (Authentication Header))*, que lida com a criptografia dos conteúdos dos pacotes. O *IPSec* é compatível com a versão *IPv6* (Internet Protocol version 6) do *IP*. Dada a lentidão na sua adopção, o *IPSec* foi adaptado para o *IPv4 (Internet Protocol version 4)* do *TPC/IP*. A Figura 6.6 mostra os cabeçalhos do *IPv4* e *IPv6* antes de ser aplicado o *IPSec*. (Martins, 2000)

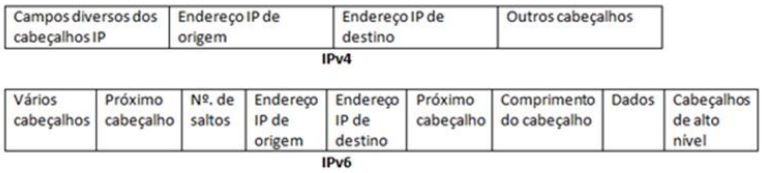

Figura 6.6 - Cabeçalhos IPv4 e IPv6 (Martins, 2000)

6.3.2 Implementações do IPSec

Nas aplicações actuais que utilizam o IPSEC este pode ser implementado e disponibilizado nos mais diversos dispositivos, como, por exemplo, servidores, *routers* ou *firewalls*. As situações mais frequentes, segundo (Dhall et al., 2012), são a implementação em Nós (*hosts*), proporcionando uma segurança "*end-to-end*", a possibilidade de implementação de todos os modos de segurança *IPSec* e garantindo ao utilizador a possibilidade de autenticação no estabelecimento de ligações *IPSec*, e a implementação em *routers*, garantindo o fluxo de pacotes entre os dois nós numa rede pública e a possibilidade de autenticar e autorizar os utilizadores a entrarem numa rede privada. O tema das redes privadas é abordado no capítulo 7 deste trabalho.

Para (Martins, 2000), o *TCP/IP* integra o IPSec nativamente na sua pilha protocolar, sendo possível implementá-lo em servidores ou gateways de segurança. Para isso recorre-se a uma implementação do tipo *BITS (Bump-In-The-Stack)*, onde o *IPSEC* é implementado entre a camada *IP* nativa e o driver de rede local existente. Não é necessário o acesso ao código fonte, tornando-se uma implementação ideal para sistemas mais antigos (*legacy systems*). Por seu lado, numa implementação *BITW (Bump-In-The-Wire)* utiliza-se um aparelho físico para a criptografia, que é endereçável na camada IP, podendo este ser encontrado em *gateways* e servidores. Quando o *BITW* é implementado num servidor único, ele é parecido com o *BITS*. Quando implementado (BITW) num *router* ou *firewall*, deve actuar como gateway de segurança.

6.3.3 Protocolos Criptográficos no IPSEC

O *IPSec* foi construído na base de determinados padrões de criptografia de forma a promover a confidencialidade, a integridade e a autenticação. Alguns dos padrões são a seguir descritos de forma sucinta: (Martins, 2000)

- Protocolo *Diffie-Hellman*, para permuta de cifra secreta entre duas partes, numa rede pública;
- Criptografia de chave pública para assinar as trocas realizadas pelo *Diffie-Hellman*, por forma a garantir as identidades das partes e assim evitar o tipo de ataque conhecido como "*men in the middle*";
- *DES*, *3DES* e outros algoritmos usados na criptografia dos dados;
- Certificados digitais, para aprovação de chaves públicas.

6.3.4 Arquitectura IPSec

A arquitectura do *IPSec* inclui vários componentes e a forma como eles interagem entre si. As 12 *RFC* que foram definidas na publicação do protocolo contêm os aspectos relevantes da arquitectura, gestão de chaves e protocolos base. (Dhall et al., 2012) A Figura 6.7 exibe as relações entre os componentes da arquitectura do IPSec.

Nesta arquitectura os elementos que tem um papel preponderante são o cabeçalho de autenticação (*AH*), o protocolo de segurança (*ESP*) e o gestor de chaves. Os dois primeiros são modulares, o que proporciona o uso de novos algoritmos. De forma a tornar num padrão os parâmetros de uma determinada transacção segura (*SA (Security Association)*), o *IPSec* usa o conceito de Domínio de Interpretação (*DI (Domain Interpretation)*), nos quais o tamanho das chaves e o formato são definidos à priori, aquando do estabelecimento da ligação segura. (Martins, 2000)

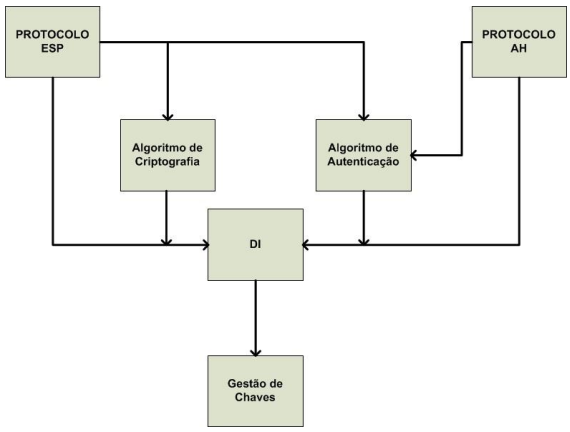

Figura 6.7 - Arquitectura *IPSec* (Martins, 2000)

Durante o processo de comunicação, e em cada fase da comunicação segura, é requerida uma *SA*. Assim sendo, para o processo de autenticação é necessária uma *SA*, e para a criptografia dos dados outra *SA*. No cenário em que se recorre ao uso do mesmo algoritmo, o conjunto de chaves é distinto. A desvantagem do uso de uma *SA* é que esta só pode ser usada na transferência de dados num sentido. Caso se pretenda uma transferência nos dois sentidos é necessária a utilização de duas *SA*. O cabeçalho de autenticação é habitualmente situado entre os campos *IP* e *TCP*, e não é realizada nenhuma alteração nos dados do pacote (*payload*). O cabeçalho *AH* apresenta cinco campos: (Martins, 2000)

- Próximo cabeçalho;
- Comprimento da parte de dados;
- Índice de parâmetros de segurança (*SPI*);
- Número sequencial de dados de autenticação;
- Verificação de integridade.

O *SPI* descreve qual o receptor do pacote e o conjunto de protocolos de segurança que o emissor utiliza. Neste caso, para a autenticação dos dados usa-se o protocolo *HMAC (Hash-based Message Authentication Code)* agrupado com o *MD5* ou o *SHA-1*. O *MD5*, após uma série de ataques de colisão, viu reduzida a sua utilização, tendo a preferência recaído na utilização do *SHA-1*, que produz um sumário de 160 *bits* (ou 20 caracteres) que está menos sujeito a ataques de colisão. (Martins, 2000)

O ataque de colisão é uma forma de ataque em que o atacante tenta encontrar duas mensagens que originam o mesmo sumário. A técnica de combinação dos protolocos *HMAC* e *SHA-1* é conhecida como *HMACSHA1 (Hash-based Message Authentication Code using Secure Hash Algorithm)*, em que o campo do autenticador tem somente 96 *bits,* sendo que o sumário é truncado no final do cálculo. O *AH* tem um mecanismo de *anti-replay* que desta forma evita retransmissões de pacotes, evitando assim a possibilidade de ataques de *DoS (Denial of Service)*. O cabeçalho *AH* é responsável apenas pela autenticação do conteúdo do pacote, sendo que o seu conteúdo transita de forma livre pela rede. Neste caso, para se assegurar a confidencialidade dos dados é necessário utilizar o cabeçalho *ESP*. (Martins, 2000)

Este é responsável pela encriptação dos dados e é incluído entre o cabeçalho *IP* e o restante datagrama. Assim os dados são modificados depois de serem encriptados. O *SPI*, que tem como função informar o recipiente do pacote de como este deverá proceder à sua abertura de forma apropriada, é também incluído. A utilização de um contador no *ESP* informa quantas vezes foi utilizado o mesmo *SPI* para o mesmo endereço *IP* de destino. A utilização deste mecanismo acautela ataques nos quais os pacotes são copiados e enviados fora da ordem, confundindo desta forma os nós de comunicação. (Martins, 2000)

O restante pacote, com a excepção da parte de autenticação, é cifrado, antes de se proceder à transmissão na rede. Os algoritmos de criptografia que têm maior uso são o *DES*, *3DES*. O *ESP* pode ser usado na autenticação, utilizando um campo opcional que é reservado a esse fim. Quando usado o *ESP* no processo de autenticação, este é diferente da autenticação proporcionada pelo *AH*, pois esta não protege o cabeçalho *IP* que antecede o *ESP*, protegendo, contudo, o cabeçalho *IP*, que é encapsulado em modo túnel. O *AH* defende este cabeçalho, simultaneamente com o conteúdo do pacote *ESP*. (Martins, 2000)

6.3.5 Modos de funcionamento do IPSEC

O funcionamento do *IPSec* ocorre em dois modos, transporte e túnel. No caso do primeiro, apenas o segmento da camada de transporte é usado, sendo que todo o pacote *IP* é autenticado ou encriptado. No caso da utilização do *IPSec* em modo de transporte, este aplica-se a implementações em servidores e *gateways*, fornecendo protecção às camadas superiores dos protocolos, bem como aos cabeçalhos *IP* escolhidos. O cabeçalho *AH* é incluído após o cabeçalho *IP* e precede os protocolos da camada superior, tais como *TCP*, *UDP* e o *ICMP*, ou antes de qualquer adição de outros cabeçalhos por parte do *IPSec*. No funcionamento do *IPSec* os endereços *IP* de origem e destino encontram-se abertos para alteração, caso seja interrompida a sua circulação por parte de um atacante. (Martins, 2000)

Quando se fala do modo túnel, somente o cabeçalho *IP* externo fica visível, dando desta forma a conhecer o seu destino ao equipamento que procede à sua transmissão (*router*, *firewall*), mas tendo o seu conteúdo interno cifrado. Mesmo utilizando o *AH* ou o *ESP* num pacote *IP*, quer no modo de transporte ou no modo túnel, o *IPSec* recorre a certas combinações dos dois modos. Utiliza desta forma o modo túnel para autenticar ou encriptar o pacote e os seus cabeçalhos, e depois aplica *AH* ou *ESP*, ou ambos, no modo de transporte, para que desta forma reforce a protecção do novo cabeçalho criado. Existe a desvantagem de que no modo túnel, não é possível utilizar o *AH* e *ESP* em simultâneo. Isto deve-se ao facto de o *ESP* ter o seu próprio esquema de autenticação. Na Figura 6.8 **Erro! A origem da referência não foi encontrada.** mostram-se os cabeçalhos do *IPSec*, nos cenários de transporte e túnel. (Martins, 2000)

Figura 6.8 - Modos Transporte e Túnel (Adaptada de (Martins, 2000))

6.3.6 Gestão de Chaves IPSEC

A gestão de chaves pelo IPSec pode ocorrer de forma manual ou automática, dependendo do número de *sites* ligados. O protocolo padrão de gestão automática de chaves usado pelo *IPSec* é o *IKE*, que recorre à combinação do *ISAKMP (Internet Security Association Management)* e do protocolo de *Oakley*. O *ISAKMP* descreve os processos para a troca de chaves. (Martins, 2000)

Segundo o autor (Berger, 2006) o *IKE* é utilizado para o estabelecimento de ligações *IPSec*, troca de chaves e partilha na autenticação dos dados. Também executa um papel no que respeita à negociação do túnel *VPN*. As mensagens do *IKE* são trocadas via protocolo *UDP* e realizadas na porta 500. Já (Martins, 2000) refere que o *IKE* opera nas duas fases, sendo que na primeira existem dois pares para criar um canal seguro e assim realizar as operações do *ISAKMP* (*ISAKMP SA*), e na segunda estes negoceiam os *SA* de âmbito geral.

O protocolo de *Oakley* promove três modos para a permuta de chaves e o estabelecimento das *SA ISAKMP*. No modo principal (*main mode*), e sendo este a primeira fase de troca do *ISAKMP*, estabelece-se uma ligação em canal seguro. O modo agressivo (*agressive mode*), outra forma de realizar a primeira fase, é mais simples e rápido que o modo principal, mas tem como desvantagem a não protecção das identidades dos nós que participam na negociação. Esta desvantagem verifica-se porque ele transmite as identidades antes do estabelecimento de um canal seguro de comunicação. O modo rápido (*quick mode*) é a segunda fase da troca, em que se negoceia um *SA* para a comunicação de uso geral. (Martins, 2000)

O *IKE* tem outro modo designado de novo grupo (*new group mode*), que não se ajusta a nenhuma das duas fases anteriormente descritas. Neste modo, recorre à fase de negociação que é usada para fornecer um mecanismo que vai definir grupos privados para execução de trocas do tipo *Diffie-Hellman*. Em seguida é descrita a maneira de estabelecer uma associação segura recorrendo ao *IKE*. O modo de negociação deve seguir os seguintes passos: (Martins, 2000)

- Algoritmo de criptografia para preservar os dados;
- Um algoritmo de sumário para a assinatura digital;
- Um método de autenticação para assinar o sumário;
- Informação acerca do grupo sobre o qual a troca *Diffie-Hellman* deve ser realizada;

Especificação da função aleatória para realizar o sumário de alguns valores durante a troca de chaves (mecanismo de verificação). Caso não seja detalhado nada, o padrão usado é a versão *HMAC*.

Na **Erro! A origem da referência não foi encontrada.** observam-se esquematicamente os elementos *IKE* e *IPSEC*, com as funções de chaves fornecidas por eles.

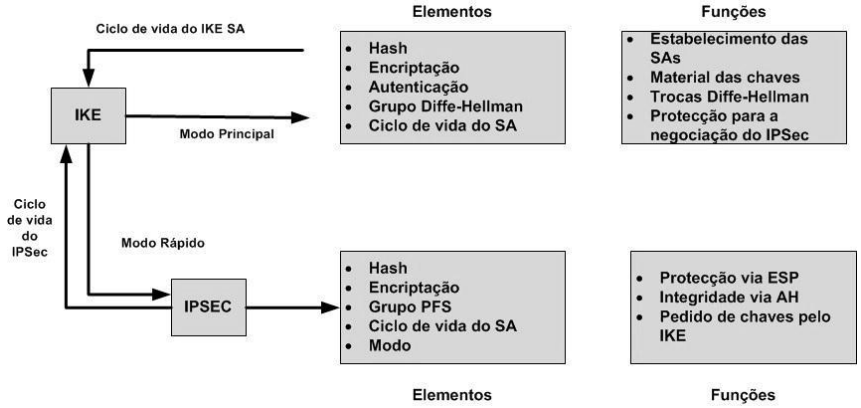

Figura 6.9 - Elementos do *IKE* e *IPSec* (Martins, 2000)

6.4 Conclusão

Neste capítulo foi abordado o tema dos protocolos de rede seguros, foi descrito o SSL, TLS e IPSec, fazendo uma introdução ao tema, suas implementações, protocolos, arquitectura, modos de funcionamento e gestão de chaves. No próximo capítulo é estudado o tema das redes privadas virtuais.

7 Rede Privadas virtuais

As Redes Privadas Virtuais, designadas por *VPN*, adquiriram uma relevância cada vez maior para os negócios das organizações. As *VPNs* permitem aos utilizadores a criação de ligações privadas entre as sedes e as filiais da sua organização, utilizando uma infraestrutura de rede pública (*Internet*). (de Rezende, 2004)

Como referido no capítulo 3, dada a massificação da Internet, a utilização de *TCP/IP* como protocolo base das comunicações na sua infraestrutura, não garante nativamente a segurança aos dados, pois este não foi desenvolvido originalmente com esse intuito. Assim, os dados transmitidos estão vulneráveis a acessos não autorizados. De forma a contornar este problema, foram testadas várias soluções, sendo a mais usada a *VPN,* apresentando um baixo custo e integrando soluções que podem recorrer tanto a *software* proprietário como livre. (Narayan et al., 2008)

As soluções *VPN* apresentam as seguintes vantagens: custo, escalabilidade e viabilidade da solução. Em seguida são abordados alguns conceitos básicos que fundamentam as *VPN*. (de Rezende, 2004)

7.1 Conceitos básicos

Como já havia sido referido, as *VPN* são um elemento de segurança, acrescentando um valor elevado quando a sua utilização é realizada num ambiente empresarial. O aspecto económico é preponderante, pois evita-se o recurso a ligações dedicadas. Assim deixa-se de parte as ligações que tradicionalmente ligavam a sede com as suas filiais bem como as suas estruturas de acesso remoto, reduzindo os custos elevados na factura das comunicações. Quando se recorre à utilização de soluções baseadas em *VPN*, tem-se como objectivo primário o uso da infraestrutura de rede pública (*Internet*). Esta é usada como *backbone*, garantindo-se assim comunicações a baixo custo entre pontos distintos. Os utilizadores remotos usam duas redes que estão fisicamente separadas, embora de forma lógica estas formem apenas uma rede. (de Rezende, 2004)

Segundo o autor (de Rezende, 2004), as *VPN* assentam em dois fundamentos básicos:

- Tunneling;
- Criptografia.

O segundo visa fornecer a uma comunicação, confidencialidade, autenticidade e integridade da ligação, sendo a base de segurança para as soluções *VPN*. Para os autores (Kuihe & Xin, 2007), um ponto crítico na utilização de *VPN* surge na encriptação e na gestão da sua cifra.

Para resolver isso recorre-se a técnicas de encriptação aritmética, validação e também a técnicas de gestão de cifras.

No caso do primeiro, este é usado para a criação de uma comunicação (virtual) que aproveita uma ligação partilhada (*Internet*) como *backbone*, garantindo o encapsulamento, transmissão e desencapsulamento dos dados na ligação estabelecida entre os dois pontos. Mesmo possuindo diversas vantagens, não se pode descurar que o uso da Internet para a transmissão de dados privados pode criar inúmeras complicações no que respeita à segurança dessa informação. (de Rezende, 2004)

Assim sendo, e para que as *VPN* garantam segurança nas suas comunicações, é imprescindível que estas sejam capazes de providenciar os seguintes requisitos que foram abordados no capítulo 4, tais como, confidencialidade, autenticidade e integridade. Assim, além destes requisitos básicos, é interessante que a tecnologia utilizada no desenvolvimento de uma *VPN* ofereça mecanismos de segurança adicionais, tais como: (de Rezende, 2004)

- Controlo de acessos;
- Restrição de acesso a utilizadores não autorizados;
- Auditorias, e outros mecanismos que proporcionem mais segurança nas comunicações das entidades envolvidas.

Na Figura 7.1 é esquematizada uma *VPN*, vulgarmente designada por *gateway-to-gateway*, as quais são transparentes para os utilizadores finais. Na mesma figura observa-se o túnel que é implementado entre dois pontos (*gateway* **A** e **B**), permitindo criar uma ligação entre dois locais da empresa geograficamente separados. O outro cenário possível numa *VPN* é o *Client-to-Gateway*. Neste cenário o túnel tem início no equipamento do utilizador, recorrendo-se à utilização de um *software* cliente *VPN*, como se pode ver na Figura 7.2. (de Rezende, 2004)

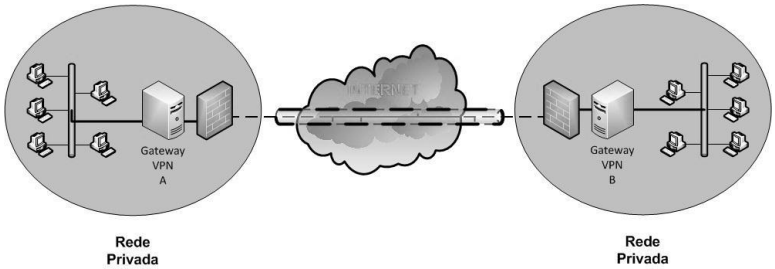

Figura 7.1 - Cenário *Gateway-to-Gateway VPN* (adaptada de (de Rezende, 2004))

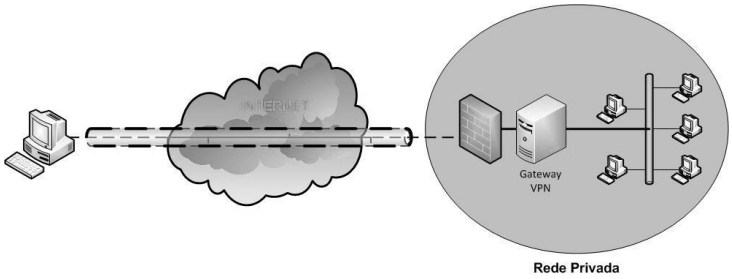

Figura 7.2 - Cenário Client-to-Gateway (adaptada de (de Rezende, 2004))

O primeiro cenário é usado na criação de uma *Intranet VPN,* garantindo-se a ligação de vários departamentos e filiais duma empresa. O segundo cenário, denominado de *Extranet VPN*, é usado nas ligações que permitem que utilizadores, parceiros, clientes ou fornecedores se liguem a uma determinada empresa. As *VPN* usadas em cenário de *Intranet* suportam as ligações de alta velocidade que predominam nas *LAN*. Crucial também é garantir a alta confiabilidade que assegura a prioridade no caso das aplicações críticas, não descurando a escalabilidade, caso seja necessário aumentar a empresa, bem como o aparecimento de novos utilizadores e aplicações.

Quando se implementa o segundo caso, a solução *Extranet*, tem de se recorrer a um padrão para que seja assegurada a interoperabilidade das várias soluções que possam existir. Nesta solução é importante garantir o controlo de tráfego, pois desta forma evitam-se congestionamentos no acesso à informação e disponibilização rápida de dados de aplicações criticas. (de Rezende, 2004)

O recurso à utilização de uma rede pública para efectuar ligações seguras entre os vários pontos de uma empresa (sede, filiais e outros agentes), representa, como referido anteriormente, menores custos, melhor flexibilidade e escalabilidade, reduzindo-se assim o impacto nos utilizadores remotos no caso de quaisquer mudanças nas suas ligações. Comparando com a utilização de ligações dedicadas, estas têm maior custo fixo mensal e, caso seja necessária alguma alteração na infra-estrutura, a importância a pagar é elevada. A Internet surge como facilitador na gestão das ligações, pois é apenas necessário um único ponto para acesso, beneficiando da ligação global dada pela *Internet*, não acontecendo o mesmo nas ligações dedicadas. Este conjunto de factores promove a interacção entre as empresas, podendo ser usado na expansão de negócios. Na Figura 7.2 é exemplificado um acesso remoto usando uma rede pública para acesso a uma *VPN*. (de Rezende, 2004)

7.2 Tunneling

O *tunneling* é um mecanismo utilizado nas *VPN* para criar uma ligação virtual. Quando esta ocorre, a sensação dos utilizadores finais é que se ligam de forma directa a uma rede privada, o que é apenas ilusório, pois na realidade é usada uma ligação sobre uma rede pública. (de Rezende, 2004)

O *tunneling* é um mecanismo nuclear numa *VPN*, pois usa um túnel que serve de meio para a transferência de dados entre duas redes distintas. (Li et al., 2009)

O mecanismo de *tunneling* não permite a circulação do pacote no seu estado original. Este é encapsulado recorrendo a um cabeçalho adicional, como exemplificado na Figura 7.3. O *tunneling* inclui, desta forma, o processo de encapsulamento, transmissão ao longo da rede pública e desencapsulamento dos pacotes. (Aqun et al., 2000)

Assim, e segundo o autor (de Rezende, 2004), o novo cabeçalho faculta informações de encaminhamento para que o pacote possa circular na rede pública, sendo os pacotes dirigidos para as extremidades do túnel. O caminho lógico criado na rede pública é chamado de túnel, e define a ligação segura entre um ponto **A** e **B**. Assim que os pacotes são recebidos no lado **B** do túnel são desencapsulados e encaminhados ao seu destino final.

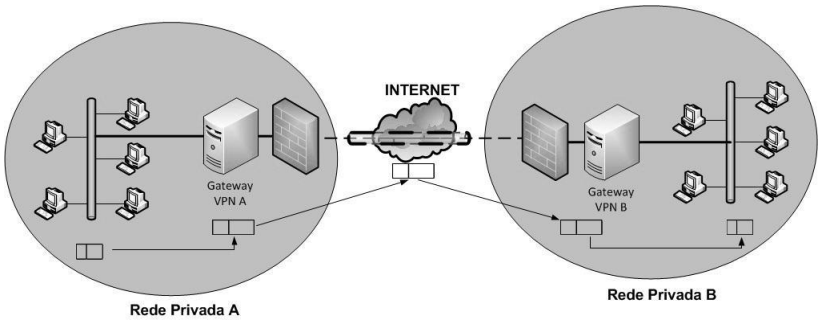

Figura 7.3 - Processo de Tunneling (adaptada de (de Rezende, 2004))

7.3 Protocolos

As *VPN* baseiam-se na criptografia e no *tunneling,* a primeira é utilizada para garantir a autenticidade, confidencialidade e a integridade das ligações, sendo por seu lado o *tunneling* responsável pelo encapsulamento e transmissão dos dados numa rede pública, entre dois pontos distintos.

A falta de mecanismos capazes de proteger o acesso remoto *VPN* acabou na especificação de diferentes padrões. Assim sendo, em determinadas situações, é possível combinar soluções no intuito de obter os serviços pretendidos. Os protocolos diferem entre si na camada do modelo *TCP/IP* onde actuam e no modo em como a criptografia é utilizada, influenciando

directamente o nível de segurança do acesso remoto *VPN*. De seguida são abordados os principais protocolos utilizados no acesso *VPN*.

7.3.1 Point-to-Point Tunneling Protocol (PPTP)

O *PPTP* foi inicialmente desenvolvido por um conjunto de empresas que se intitularam de *PPTP* Fórum, formado pela *3COM*, *Ascend Communications*, *Microsoft*, *ECI Telematics* e *US Robotics*. O objectivo do *PPTP* era o de repartir as funções do acesso, possibilitando assim que qualquer utilizador utilizasse a rede pública (*Internet*) de forma a estabelecer uma ligação segura entre o seu equipamento e uma rede privada. O *PPTP* usa a técnica de *tunneling* de tráfego *PPP* nas redes *IP*. (de Rezende, 2004)

Figura 7.4 - Encapsulamento de um datagrama IP feito pelo PPTP (adaptada de (de Rezende, 2004))

No decorrer do processo e previamente ao envio de um datagrama *IP*, o *PPTP* realiza duas operações: em primeiro lugar cifra e depois encapsula o datagrama num pacote *PPP*. Por fim, é encapsulado num pacote *GRE (Generic Routing Encapsulation)*, como demonstrado na Figura 7.4. (de Rezende, 2004)

De forma a garantir uma ligação segura, o *PPTP* inclui três fases, cada uma das quais tem de estar terminada antes de poder passar à seguinte. (Narayan et al., 2008)

Tal como os restantes protocolos de segurança, o *PPTP* pede a negociação de parâmetros, e só após este processo é que vai proteger verdadeiramente o tipo de tráfego entre dois pontos. Salienta-se que o seu processo de negociação é efectuado sem recurso a qualquer tipo de protecção, criando uma janela de oportunidade para que um qualquer atacante realize um ataque de modificação ou mesmo a aquisição de dados como: (de Rezende, 2004)

- O endereço *IP* dos equipamentos localizados nas extremidades do túnel;
- O nome e versão do *software* usado;
- O nome do utilizador e em certas situações o sumário da palavra passe do utilizador.

Uma das falhas apontadas ao *PPTP* é a de que as mensagens do canal de controlo circulam livremente sem qualquer tipo de autenticação ou protecção de integridade, possibilitando que esse canal de controlo fique exposto a um possível "roubo de ligação", sendo possível ao atacante introduzir falsas mensagens de controlo ou mesmo alterar as originais, por vezes sem qualquer possibilidade de detecção. Outra falha deste protocolo reside no facto de o cliente só realizar a sua autenticação no final do processo de criação dos parâmetros, expondo o equipamento a ataques do tipo *DoS (Denail of Services)* que podem levar à saturação da rede e paragem dos serviços disponibilizados. (de Rezende, 2004)

O *PPTP* é capaz de suportar inúmeros protocolos, como o *IP*, e outros, existindo a possibilidade do encapsulamento dos seus pacotes em pacotes *PPP*. Após isso, os pacotes são encapsulados recorrendo ao mecanismo de *tunneling* do protocolo *PPTP*. Findo este processo, os mesmos são encapsulados em pacotes *IP* prontos a serem transmitidos. (Wu, 2009)

7.3.1.1 A Microsoft e O PPTP

Neste ponto é descrito o *PPTP* da *Microsoft*, sendo que esta versão possui extensões proprietárias incluídas nos seus sistemas operativos *Microsoft Windows*. Estes sistemas têm uma enorme implementação, quer no mercado empresarial, quer no mercado doméstico. Um dos pontos mais débeis desta implementação encontra-se nos formatos de armazenamento e transmissão dos sumários e das palavras-chave, conhecida por *LanMan*. Um exemplo é o caso do sistema *Microsoft Windows NT* em que as palavras-chave possuem 14 caracteres, e que quando são armazenadas no formato *LanMan*, que é "*case-insensitive*", todos os seus caracteres são transformados em maiúsculas, reduzindo assim o número de sumários possíveis e tornando mais fácil os ataques por força bruta. (de Rezende, 2004)

Ainda assim a grande vulnerabilidade do *LanMan* reside na divisão da cadeia que inclui os 14 caracteres, em duas cadeias de 7 caracteres. Como os algoritmos de sumário são gerados de forma separada, um para cada uma das cadeias de 7 caracteres, o esforço necessário num ataque de força bruta é reduzido. Outra vulnerabilidade reside no facto dos parâmetros de negociação do *PPTP* serem transmitidos sem protecção, pelo que o atacante pode comprometer o sumário de uma palavra-chave que seja armazenada no formato *LanMan*. (de Rezende, 2004)

Um dos factores que pesa de forma negativa numa grande maioria de utilizadores é que estes usam palavras-chave previsíveis, tais como nomes ou objectos, as quais são facilmente quebradas por ataques do tipo dicionário. Este tipo de ataques consiste em utilizar palavras que constam num ficheiro que contem palavras comuns, tal qual um dicionário, e o seu sucesso está ligado ao facto da palavra-chave constar nessa lista. O uso do *PPTP* da *Microsoft* expõe de forma insegura as palavras-chave dos utilizadores dos sistemas operativos *Windows*. Outra vulnerabilidade grave da implementação *PPTP* em sistemas *Microsoft* é o tamanho e o processo de geração de chaves encriptadas. São usados dois modos de confidencialidade, que são disponibilizados pelo algoritmo *RC4*, sendo que um recorre a chaves de 40 *bits* e outro de

128 *bits*. No caso do primeiro, o uso de chaves pequenas permite ataques de força bruta. No caso do segundo, mesmo conseguindo uma maior segurança devido ao tamanho da chave, a utilização repetida da mesma palavra-chave reduz o número de tentativas que será necessário usar no ataque. O cenário torna-se ainda mais preocupante caso o atacante obtenha a palavra-chave do utilizador no formato *LanMan*. Estes tipos de ataques podem ser bem-sucedidos em poucos minutos. Expostas estas vulnerabilidades do *PPTP* em ambientes *Windows*, a própria *Microsoft* recomendou a desabilitação do *LanMan* e o recurso a outras soluções. (de Rezende, 2004)

7.3.2 Layer Two Tunneling Protocol (L2TP)

O *Layer 2 Tunneling Protocol (L2PT)* foi desenvolvido recorrendo ao *Layer 2 Forwarding (L2F)* e *PPTP*, sendo o seu principal objectivo o encapsulamento de pacotes *PPP*. O *L2PT* é usado abaixo da camada *IP*. No que respeita à segurança, realiza o encapsulamento de pacotes *PPP* e recorre a mecanismos de autenticação do *PPP* como o *Encryption Control Protocol (ECP)* e o *Compression Control Protocol (CCP)*. (de Rezende, 2004)

O protocolo *L2PT* suporta a autenticação do túnel, possibilitando que ambos os pontos da ligação sejam autenticados. Mesmo assim o *L2PT* não possui mecanismos robustos para protecção do túnel *L2PT*, ficando os pacotes de dados de controlo vulneráveis a ataques. Podem desta forma ocorrer ataques do tipo de modificação, *DoS* (negação de serviço). Por norma estes ataques podem resultar na interrupção ou enfraquecimento do processo de autenticação, tornando assim possível a obtenção das palavras-chave dos utilizadores. Face a estes problemas de segurança, a sua utilização em redes abertas (*Internet*) deve ser sempre efectuada em conjunto com protocolos que garantam a sua segurança. (de Rezende, 2004)

7.4 Conclusão

No capítulo das redes privadas virtuais foram introduzidos conceitos básicos e cenários de VPN. Foi abordado o tema do *tunneling* e protocolos tais como o PPTP e L2TP. Foi também estudado o PPTP em ambientes Microsoft. No próximo capítulo é descrita a implementação da solução prática deste trabalho.

8 Descrição da Implementação

Neste capítulo é descrita as soluções e a arquitectura implementadas, e em seguida uma descrição sucinta do processo de implementação.

8.1 Arquitectura da Solução

Nos capítulos anteriores foram discutidos vários temas sobre a segurança em redes. No que respeita aos laboratórios remotos, para a solução apresentada neste trabalho optou-se por uma ligação *VPN* recorrendo a uma solução em código aberto, o *pfSense*.

Esta solução permitiu a implementação de três tecnologias diferentes em *VPN* para laboratórios remotos. As soluções abordadas foram o *OpenVPN*, *PPTP* e *IPSec*. O objectivo do estudo foi garantir a existência de canais seguros que protejam as ligações a laboratórios remotos. Na Figura 8.1 pode-se ver a solução implementada para qualquer um dos casos.

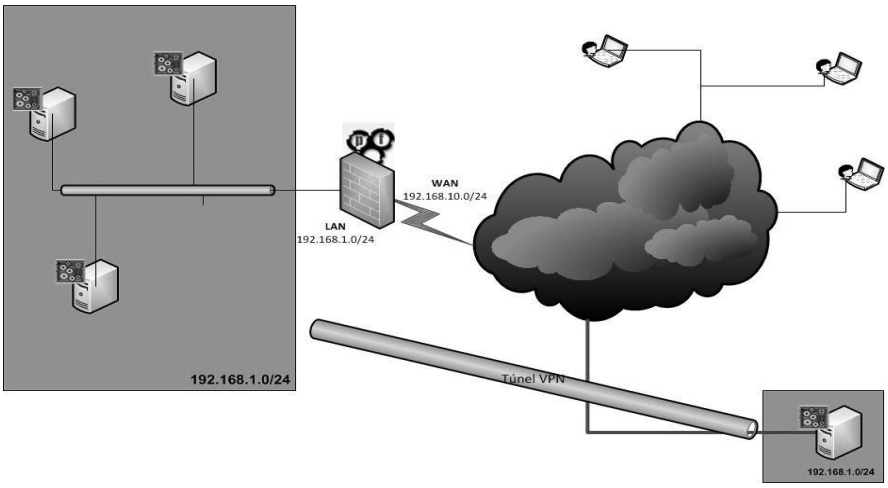

Figura 8.1 - Arquitectura da solução proposta

Neste cenário verifica-se a existência de um laboratório remoto que está fora da *LAN* 192.168.1.0/24 e que possibilita ligações remotas de forma segura a esta rede privada. Este laboratório pode ser operado pelos utilizadores que estabelecem a ligação através da *firewall*.

8.2 *pfSense* - A Escolha

Neste trabalho optou-se por uma solução de código aberto de forma a minimizar os custos, que é um dos objectivos para qualquer organização. A escolha do *pfSense* foi feita com base na possibilidade da sua personalização e adaptação aos vários cenários estudados neste trabalho, pois suporta nativamente todas as tecnologias abordadas (*OpenVPN*, *PPTP*, *IPSec*).

O *pfSense* é uma *firewall* de código aberto que se baseia em *FreeBSD* e que é totalmente personalizável, podendo, além da vertente *firewall*, ser também usada como *router*. O *pfSense* inclui além da *firewall* e do *router*, uma longa lista de pacotes que permitem a sua expansão sem pôr em causa o seu funcionamento e sem acrescentar qualquer tipo de vulnerabilidades à solução base. As implementações vão desde pequenas redes domésticas até redes de grande escala, onde se incluem grandes organizações, como universidades, permitindo a protecção de centenas ou mesmo milhares de dispositivos.

O projecto do *pfSense* teve início em 2004 e surgiu como variante de um outro projecto chamado *m0n0wall*, sendo que o *pfSense* é mais focado para a instalação em computadores ao invés do *m0n0wall* que está focado para sistemas de *hardware* integrados.

Tal como foi dito no parágrafo anterior, o *m0n0wall* é um projecto que surgiu com o objectivo de criar um sistema embutido completo, juntando-se uma *firewall* a um sistema operativo embutido, fornecendo características semelhantes às *firewall* comerciais, com a vantagem de ter um custo menor, visto que usa software de código aberto. Este projecto é baseado numa versão mais optimizada do *FreeBSD*, à qual se junta um servidor *web*, *PHP* entre outros utilitários. (Kasper, 2011)

O pfSense disponibiliza um conjunto de funções, tais como:

- Firewall;
- Network Address Translation (NAT);
- Redundância;
- Balanceamento de Carga;
- VPN;
- IPsec;
- OpenVPN;

- PPTP;
- Servidor *PPPoE*;
- Monitorização e Relatórios;
- Gráficos RRD (round robin database);
- Informação em tempo real;
- *DNS* dinâmico;
- OpenDNS;
- ZoneEdit;
- Portal de Captura;
- Redirecção de *URLs*;
- Filtragem por endereço *MAC*;
- Autenticação *RADIUS*;
- Servidor e relé de *DHCP*.

8.3 *pfSense* - Alternativas

A escolha do pfSense deveu-se ao facto de este apresentar uma arquitectura baseada em *FreeBSD*, um sistema conhecido pela sua robustez no que respeita à segurança. A existência de conhecimento de utilização prévia do *pfSense* e a existência de documentação foram também argumentos que contribuíram para esta escolha. Como foi referido, a utilização do *FreeBSD* proporciona uma maior robustez, performance e segurança, com a desvantagem de ter menos suporte de *hardware* que as distribuições de *Linux* mais comerciais. Por isso é necessária uma melhor selecção na escolha do *hardware*, como, por exemplo, a escolha das placas de rede.

Quando é realizada uma escolha deve-se ter sempre em conta a existência de alternativas, sendo que uma das alternativas a esta escolha era a utilização de soluções proprietárias com *hardware* específico, como, por exemplo, soluções da *Cisco*. Esta alternativa não foi considerada por apresentar um maior custo, dado a utilização de *software* e *hardware* proprietário, sendo também de uma maior complexidade a nível da infra-estrutura e com menor oferta de software a nível das máquinas cliente.

Podia ter-se recorrido igualmente à utilização de soluções com *hardware* genérico e *software* comercial, como na alternativa anterior, mas este também apresenta um maior custo em relação à solução escolhida e poucas alternativas no que respeita ao *software* de suporte e cliente.

Uma outra alternativa que apresenta um custo baixo, é o caso da utilização de uma distribuição comercial de *Linux*. Esta, no entanto, apresenta um esforço enorme (tempo) no que respeita à sua configuração, devido ao facto de ter de ser realizada em modo texto. Como no caso da alternativa anterior, podia recorrer-se a uma distribuição *Linux* que utilizasse para isso o modo gráfico, mas esta alternativa exige mais recursos, sendo que as máquinas a utilizar teriam que disponibilizar um *hardware* com características mais potentes que as características necessárias na solução *pfSense*. Neste cenário com ambientes gráficos, pode existir outro tipo de utilização e desta forma um maior consumo de recursos da máquina o que levará a uma menor performance da solução.

Assim, e após a consideração dos pós e contras, a escolha recaíu sobre o *pfSense*. Como foi referido anteriormente, é uma solução de baixo custo *(open source)* que consome menos recursos em termos de *hardware*, permitindo a reutilização de *hardware* mais antigo.

Disponibiliza nativamente uma interface gráfica, e caso se queira, também oferece a interface em linha de comando. O processo de actualização é simplificado, apenas requerendo um *download*, e todo o processo de actualização é automático e rápido e corre numa máquina independente que apenas corre a solução *pfSense*.

8.3.1 Tipos de firewall

Como é do conhecimento geral, uma *firewall* tem normalmente funções de encaminhamento. Esta obedece a um conjunto de regras que normalmente estão definidas num ficheiro de texto. Este funcionamento básico e o mais poderoso. A utilização das regras através de um ficheiro de texto apenas requer um editor de texto, como, por exemplo, em *Linux* o *vi*. Como exemplo deste tipo de *firewalls* tem-se o caso do *ipchains* (sistema descontinuado a partir do *kernel* 2.2), do *iptables* e do *pf*, este último usado em ambientes *OpenBSD, NetBSD* e *FreeBSD*.

Por forma a contornar a dificuldade para o utilizador menos experientes em modo texto, começaram a surgir as interfaces gráficas, normalmente implementadas recorrendo a servidores *Web* para proceder à sua configuração. Por fim, existem sistemas em que a integração e adaptação já foram realizadas, funcionando como equipamentos dedicados. Desses, são exemplos, o *m0n0wall,* o *pfSense,* ou o *ipcop*, entre outros.

8.4 Portas de rede

O tema das portas de rede já foi abordado no capítulo 3. Basicamente, a combinação de uma porta com um endereço IP é suficiente para indicar a origem e o destino exacto para que se

possa realizar a comunicação. Desta forma, uma máquina consegue aceitar dados de múltiplas fontes e de várias aplicações. Esta operação decorre em simultâneo, pois quando se articula uma porta de origem a um endereço IP com uma porta de destino e um IP de destino, já se especificou a ligação entre os processos de comunicação. (Loshin, 2004)

Os protocolos da camada de transporte identificam a origem e o destino, da mesma forma que no mundo real se identifica um apartamento ou uma casa, onde se pretende realizar a entrega do correio, sendo os processos especificados na camada mais baixa. Existem *hosts* que oferecem os seus serviços para escutar certas portas aquando da solicitação de serviços. Quando se recebe uma mensagem de forma imprevisível em certas portas, esta é transmitida a um processo associado a essa porta, processos esses conhecidos por *daemons*, pois recebem um pedido e iniciam um novo processo para lidar com o pedido, ficando após isso à escuta de novos pedidos. O *TCP* e o *UDP* usam números de portas com um endereçamento de 16 *bits*, sendo que qualquer valor entre o e 65535 é valido, sendo a porta 0 é um endereço reservado. (Loshin, 2004)

Os números de portas estão agrupados em: (Loshin, 2004)

- Portas conhecidas;
- Portas registadas;
- Portas dinâmicas (também conhecidas como privadas).

Existem dois tipos de portas, que por vezes são referidas como sinónimos: são o caso das portas efémeras e das portas transitórias, não sendo consensual qual o estado e os limites que definem este tipo de portas. Segundo o autor (Loshin, 2004), as portas são definidas da seguinte forma:

- Portas conhecidas, são definidas pela *IANA*, e são as portas que a maioria dos sistemas utiliza no seu sistema (raiz). Também podem ser usadas por processos ou utilizadores com privilégios. Por norma utilizam-se esta portas para se correr serviços conhecidos, como por exemplo *FTP*, *SMTP* entre outros.
- Portas registadas, também são definidas pela *IANA* mas não lhes é atribuída uma listagem. São normalmente usadas por serviços muito específicos, definidos pelos utilizadores ou aplicações.
- Portas dinâmicas / privadas, não existe qualquer restrição à utilização destas portas. Podem ser usadas para qualquer fim.
- Portas Efémeras / Transitórias, como os seus nomes indicam, são utilizadas pelo serviço quando necessárias, não sendo mantidas de sessão para sessão. A utilização destas portas pode acontecer quando um qualquer *host* pretende uma ligação a um serviço / porta conhecido e lhe é atribuída uma porta transitória.

O *ITEF* através da *IANA* atribuiu uma gama às portas dinâmicas entre o valor 49152 à 65535. Claro que estes valores estão sempre restringidos aos valores definidos numa *firewall*, visto que estes equipamentos podem restringir a entrega de pacotes em qualquer porta. Muitas das vezes recorre-se a este meio para restringir ataques. (Loshin, 2004)

O controlo das portas de rede permite aos administradores de rede restringir o uso de pontos de acesso a serviços (portas) para proteger a comunicação com sistemas autenticados e autorizados. (IEEE, 2010)

As entidades do sistema e opções implementadas para uma determinada porta, podem depender da porta de rede que é utilizada por determinadas aplicações de controlo de acesso ao sistema. (IEEE, 2010)

Numa comunicação é necessário garantir a autenticação e autorização por parte dos elementos que participam no processo de comunicação, sendo que desta forma se garante que a mesma é realizada apenas na porta autorizada, permitindo-se assim a comunicação entre os elementos da rede. (IEEE, 2010)

Na Tabela 1 é apresenta uma lista das portas de rede mais utilizadas.

Tabela 1 - Registo de portas por serviço (Fonte: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml)

Service Name	*Port Number*	*Transport Protocol*	*Description*
ftp	21	tcp / udp	File Transfer
ssh	22	tcp / udp	The Secure Shell (SSH) Protocol
telnet	23	tcp / udp	Telnet
smtp	25	tcp / udp	Simple Mail Transfer
http	80	tcp / udp	World Wide Web HTTP

8.5 Implementação da Solução

Neste ponto é descrito o processo que conduziu à implementação das várias soluções, livres e de baixo custo, de forma a garantir segurança nas comunicações com os laboratórios remotos. São descritas as instalações da própria *firewall* (*pfSense*) e todas as tecnologias *VPN* usadas durante a fase de estudo. Por último, é apresentada uma pequena descrição das dificuldades encontradas no processo de implementação das soluções propostas.

8.5.1 Instalação do pfSense

A primeira instalação realizada no decorrer deste trabalho foi a própria *firewall*, sendo a versão instalada a *pfSense* 2.0 (i386), obtida no site http://www.pfSense.org. Esta instalação foi realizada numa máquina física que apresenta as seguintes características:

- Processador – Intel Pentium III 650 MHz;
- Memória – 384 MB;
- Disco – 10 GB.

O processo de instalação da *firewall pfSense* é rápido (cerca de 10 minutos), sendo que depois de iniciado o sistema apenas se tem que definir as interfaces de rede e o processo está completo. (ver *Anexo I – Instalação de firewall pfSense*)

As palavras-chave utilizadas neste trabalho são fictícias, numa implementação num ambiente de produção são aconselhadas palavras-chaves com um nível de complexidade ajustados à realidade. Para este trabalho foram definidas as interfaces **RL0** para *WAN* (192.168.10.252) e **RL1** para *LAN* com o endereço (192.168.1.1). No final deste processo de instalação base parte-se para a configuração da *firewall* através do seu portal, processo que permite definir o nome, domínio, e demais opções.

Para se executar o processo de configuração só é necessário estar ligado directamente à *firewall* através de um cabo de rede e colocar no *browser* o seguinte endereço http://192.168.1.1. O processo inicia-se através de um configurador. Na primeira vez os dados de autenticação que são introduzidos são os que vêm por omissão:

- Utilizador: *admin*
- Palavra-chave: *pfSense*

De seguida, define-se o nome do *host* (neste caso foi definido como *pfSense*), seguido do nome de domínio, que é *laboris.isep.ipp.pt*. Define-se o *DNS* primário que será o endereço 192.168.10.1, sendo que no mesmo configurador se define também o servidor de hora (neste caso foi deixado o servidor por omissão).

No passo seguinte é definida a componente *WAN*, existindo várias opções, entre as quais:

- Endereço fixo;
- MAC Address;
- Servidor de *DHCP*;
- PPPoE;
- PPTP;
- Bloqueio de redes.

No menu seguinte é definida a interface de rede local, sendo que neste trabalho foi usado o endereço 192.168.1.1/24. Durante o processo de configuração inicial foi alterada a palavra-chave de administração para o valor: **1234**.

Com o passo anterior conclui-se a configuração da *firewall*, estando esta pronta para a utilização. Este processo de configuração pode ser observado no *Anexo II – Configuração inicial da firewall pfSense*. No próximo ponto será abordado a primeira tecnologia utilizada.

8.5.2 Implementação OpenVPN

Neste parágrafo é introduzido o *OpenVPN,* que é um *software* de código aberto que permite aos utilizadores criarem redes privadas virtuais *(VPN)* do tipo ponto-a-ponto ou servidor-"multi-cliente", usando túneis encriptados entre os pontos da *VPN*, sendo possível estabelecer ligações directas, mesmo estando atrás de uma *NAT firewall*, sem necessidade de reconfigurar a rede. O *OpenVPN* permite vários tipos de autenticação, sendo disponibilizadas a autenticação através de chaves partilhadas, de certificados digitais ou a autenticação com utilizador e palavra-chave.

O *OpenVPN* utiliza extensivamente a biblioteca *OpenSSL*, mas também faz recurso a protocolos como o *SSL* (v3) e o *TLS* (v1), encontrando-se disponível para clientes *Solaris*, *Linux*, *OpenBSD*, *FreeBSD*, *NetBSD*, *Mac OS X* e *Microsoft Windows*. O *OpenVPN* não é um cliente baseado na *Web* e não é compatível com o *IPSec* ou qualquer outro tipo de *VPN*. Os pacotes *OpenVPN* consistem num binário que é utilizado quer no lado do cliente, quer no lado do servidor, não sendo compatíveis com outras soluções *VPN*, como por exemplo *IPSec*.

Assim, o objectivo da implementação assente em *OpenVPN* é permitir a ligação de uma ou mais máquinas através de uma *VPN*. Neste trabalho restringiu-se apenas a cinco clientes o número de ligações aceites. Recorreu-se à autenticação com um utilizador *(vpnuser_ov)* e palavra-chave (*1234*), procedendo-se à configuração de dois certificados digitais, um *CA* interno e um *CA* de servidor, de forma a tornar a ligação mais segura. A configuração do lado da *firewall*, como foi referido anteriormente, tem um certificado de raiz interno *(ISEP Laboris Thesis CA)*, que possui uma chave de 2048 *bits* e um período de vida de 365 dias.

Após a criação dos certificados, foi criado um utilizador *(vpnuser_ov)* e uma palavra-chave *(1234)*, o qual foi associado ao certificado *ISEP Laboris Thesis CA*. O certificado de servidor foi chamado de *ISEP Laboris Server CA* e tem uma chave de 2048 *bits* e um período de vida de 3650 dias (aproximadamente 10 anos). De seguida, criou-se uma ligação usando o *UDP*, com recurso à porta 1194, que é a porta por omissão do *OpenVPN*, segundo a numeração oficial da *IANA (Internet Assigned Numbers Authority)*. O uso do *UDP* em detrimento do *TCP* é motivado pelo facto de o *UDP* ser um protocolo por omissão do *OpenVPN*. O *OpenVPN* multiplexa toda a comunicação em cima de uma única porta *UDP* e tem a facilidade de conseguir trabalhar com a maioria dos servidores *proxy*, incluindo *HTTP*, e consegue, como já foi referido, passar nas *firewalls* que utilizam *NAT*.

No *OpenVPN* são disponibilizadas dois tipos de interfaces: o *TUN/TAP*. Neste trabalho foi utilizado o *TUN* que permitiu criar um túnel na camada 3, possibilitando a passagem pela *firewall* sem problemas. Por sua vez o *TAP* cria um túnel na camada 2, permitindo qualquer tipo de tráfego sobre *Ethernet*.

Como referido no parágrafo anterior, existem dois tipos básicos de interface de rede virtual: o *TUN* e o *TAP*, sendo conhecidos por *"TUN/TAP" driver*. As interfaces *TUN* são ponto-a-ponto, pelo que a rede na qual uma dessas interfaces participa assemelha-se a uma ligação ponto-a-ponto, como, por exemplo, uma rede feita com um cabo cruzado, pois nela só participam dois componentes. Quando se utiliza este tipo de interface *TUN*, a aplicação vai lidar directamente com pacotes *IP*, sendo que a interface da rede virtual é configurada via comando *ifconfig* com o endereço *IP* da nossa máquina e da máquina remota, como se pode observar no esquema apresentado na Figura 8.2. (Matias, 2007)

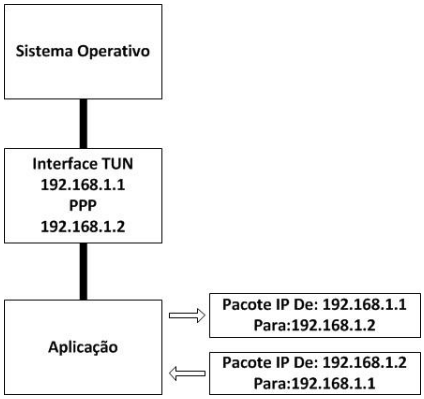

Figura 8.2 - Interface *TUN* (adaptada de (Matias, 2007))

No caso das interfaces *TAP*, estas actuam como sendo uma placa de rede ligada a um comutador, pois participam numa rede com vários componentes, cada um identificado por um endereço *MAC*. Desta forma, este tipo de interface inclui um encapsulamento *Ethernet*, abaixo do qual serão carregados pacotes de protocolos como *IP* e *ARP*. A interface de rede virtual é configurada via *ifconfig* com um endereço *MAC*, e, caso seja pretendido, também com um endereço *IP*. Na Figura 8.3 é dado o exemplo de uma interface *TAP* com endereço *MAC*. (Matias, 2007)

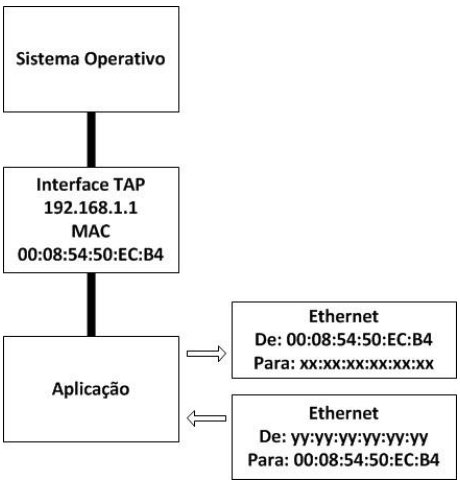

Figura 8.3 - Interface *TAP* (adaptada de (Matias, 2007))

A biblioteca de compressão *LZO (Lempel-Ziv-Oberhume)* foi utilizada para compactar o fluxo de dados. Nas versões mais antigas de *OpenVPN* existia uma restrição, que foi abolida na versão utilizada neste trabalho, e que impossibilitava que um processo pudesse manipular e gerir vários túneis em simultâneo. O uso do *TLS* permite a criação de túneis entre entidades, os quais têm como características a autenticação, integridade e não-repúdio através de certificados digitais. A confidencialidade é assegurada através de algoritmos de cifra, *Blowfish*, *AES*. É igualmente permitida a compressão da informação antes da cifra. O *TLS* tem uma fase inicial de negociação em que são trocadas informações (cifras utilizadas, compressão (sim ou não), certificados digitais, entre outros) de forma a estabelecer o túnel. Nesta implementação foi usada o *BF-CBC (Blowfish-Cipher Block Chaining)*.

O *OpenVPN* utiliza UDP, o que é uma vantagem relativamente ao *IPSec,* pois em algumas situações os *ISP (Internet Service Provider)* bloqueiam protocolos específicos para *VPN*, tornando o uso de *VPN* impossível a não ser com soluções específicas fornecidas à medida, que se tornam dispendiosas. Em suma, podemos afirmar que as vantagens do uso de *OpenVPN* são:

- Menor custo com as ligações físicas;
- Menor custo administrativo em redes complexas;
- Maior alcance e abrangência da solução;
- Maior rapidez na implementação;
- Melhor escalabilidade dos recursos (equipamentos);
- Maior controlo no tráfego de dados.

Algumas desvantagens das soluções *OpenVPN* podem igualmente ser enumeradas, tais como:

- Latência alta devido aos processos criptográficos;
- Falta de compatibilidade com IPSec;

No que respeita a vantagens da utilização do *OpenVPN* em relação ao *IPSec*:

- Grande facilidade de instalação e configuração;
- Excelente desempenho perante *Firewall NAT*, nos filtros de pacotes, evitando a necessidade de adaptações e de modificações no sistema de *firewall*;
- Permite o uso de UDP, evitando o efeito "*TCP over TCP*";
- Utilização de chaves assimétricas para a autenticação;
- Disponível para Linux, MS Windows e MAC OS X;
- Grande estabilidade.

8.5.2.1 Problemática do "*TCP over TCP*"

Neste ponto será abordado um problema que ocorre nas *VPNs*, que é o *"TCP over TCP"*. Uma ideia frequente nas aplicações de *tunneling IP* é a aplicação correr como o protocolo *PPP*, que encapsula os pacotes *IP* num formato que permite um transporte de fluxo (como uma linha de *modem*), sobre uma conexão *TCP*. Isto seria uma solução fácil para a encriptação de túneis *PPP* sobre *SSH*. (Titz, 2001)

Seria também um modo de compressão fácil para o tráfego *IP*, não fosse a compressão baseada no datagrama obrigar à difícil tarefa de superar os limites da eficiência. Infelizmente esta ideia não funciona muito bem, pois é frequente ter atrasos muito longos e ligações

frequentemente abortadas, e isto deve-se ao algoritmo de retransmissão *TCP*. O *TCP* divide o fluxo de dados em dois segmentos, que são enviados individualmente como dois datagramas *IP*. Os segmentos transportam um número de sequência, que numera os *bytes* no fluxo, e um número de *acknowledge*, que indica à outra parte o ultimo número de sequência recebido. (Titz, 2001)

Como os datagramas *IP* podem ser perdidos, duplicados ou reordenados, os números de sequência são usados para reconstruir o fluxo de dados. O número de *acknowledge* diz ao emissor indirectamente, se um segmento foi perdido, se quando esse *ACK* acabado de ser enviado não é recebido num certo período de tempo, situação na qual o emissor dá o pacote como perdido e volta a enviá-lo. Existem muitos protocolos que têm uma abordagem semelhante e que são normalmente usados em ligações onde existe uma largura de banda relativamente fixa. No entanto, num cenário como o da *Internet*, em que esses parâmetros (largura de banda, atraso e perda de taxa) não são controláveis e variam de ligação para ligação, um *timeout* fixo não é o método mais apropriado, sendo que este mesmo *timeout* fixo também não é apropriado mesmo em redes *LAN*. Assim sendo, este método de *timeout* fixo aumenta o congestionamento da rede e leva a um efeito conhecido como fusão (*meltdown*). (Titz, 2001)

Por este motivo, o TCP usa *timeout* adaptativo para todos os parâmetros de temporização. Os *timeout* começam de forma conservadora e vão se alterando dinamicamente à chegada de novos segmentos. Assim, com esta forma de adaptação, existe a possibilidade de mudança de forma a evitar o efeito de fusão. As políticas de *timeout* do *TCP* funcionam bem para cenários como a *Internet*, pois o *TCP* tenta não perder as ligações, aumentando para isso o *timeout* em vários minutos. Isto é aplicado sensatamente em cenários onde se verifica transferência de dados em massa realizados de forma autónoma. (Titz, 2001)

Esta optimização para as quebras de fiabilidade verifica-se quando se tenta agregar uma ligação *TCP* em cima da outra. Este cenário nunca foi antecipado pelos criadores do *TCP*, mas acontece quando temos *PPP* a correr sobre *SSH* ou outro protocolo baseado em *TCP*. Isto deve-se ao facto do *PPP* encapsular datagramas *IP*, e estes serem transportados como *TCP payload*, podendo-se observar esse esquema na Figura 8.4. (Titz, 2001)

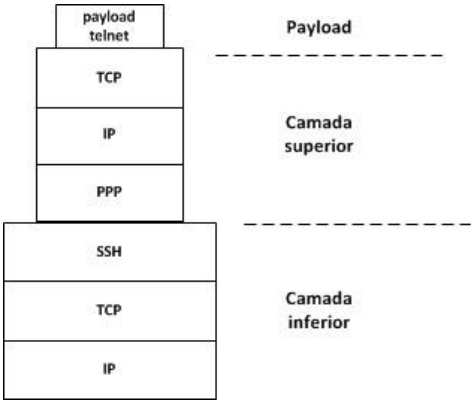

Figura 8.4 - *TCP payload* (adaptada de (Titz, 2001))

É de notar que tanto a camada superior como a inferior do *TCP* tem temporizadores diferentes, pois verifica-se que quando a camada alta do *TCP* começa rápida, os seus temporizadores também são rápidos. No entanto, pode acontecer que em ligações de baixo nível, os seus temporizadores sejam lentos, talvez como um resquício de um período de uma ligação lenta ou não fiável. (Titz, 2001)

Podemos imaginar o que é começar a perder pacotes caso a situação acima se observe, caso em que as filas da camada baixa do *TCP* começam a gerar retransmissões e aumentar os *timeout*. Após a comunicação ser bloqueada por causa de um grande *timeout*, a camada superior do *TCP* não obtém em tempo útil um *ACK*, e assim coloca em fila uma retransmissão. (Titz, 2001)

Observa-se ainda que se o *timeout* da camada superior do *TCP* é inferior ao *timeout* verificado na camada inferior, isto leva a que camada superior coloque mais retransmissões em fila, sendo que estas não conseguem ser processadas pela camada inferior, o que provoca que a camada superior pare e que cada retransmissão acrescente mais um problema à comunicação, verificando-se assim o efeito de fusão. (Titz, 2001)

Assim sendo, as provisões de fiabilidade do *TCP* são postas em causa neste ponto, pois as retransmissões da camada superior do *TCP* são desnecessárias, uma vez que o transporte garante a entrega da informação. O problema é que a camada superior não sabe disto, visto que o *TCP* assume sempre um transporte não fiável. (Titz, 2001)

No ponto seguinte será abordada a segunda tecnologia usada para a criação de *VPN*, o *PPTP*. Todas as configurações do *OpenVPN* podem ser consultadas no *Anexo III - Configuração do OpenVPN na firewall pfSense* e *Anexo IV - Configuração do OpenVPN no Windows* e *Linux*. (Titz, 2001)

8.5.3 Implementação PPTP VPN

O *PPTP* é um protocolo que foi originalmente desenvolvido por um grupo de empresas que se denominaram de *PPTP Fórum*, constituído pela *3COM*, *Ascend Communications*, *Microsoft, ECI Telematics e US Robotics*. No início a ideia base do *PPTP* era dividir as funções de acesso remoto, de tal modo que qualquer utilizador pudesse usar a infra-estrutura da *Internet* para conseguir uma comunicação segura entre clientes remotos e redes privadas. No capítulo 7 pode-se obter mais informação sobre o *PPTP*.

No cenário de implementação da *VPN* recorrendo à tecnologia PPTP foi definido um endereço de um servidor que, no caso deste estudo, foi o 192.168.10.1, o qual também funciona como *gateway*. Este endereço terá de ser um endereço diferente dos endereços que são usados nas interfaces de rede *WAN* e *LAN*.

Foi também necessário definir o ponto inicial da gama de endereços para a rede remota, sendo que neste caso foi definido o primeiro endereço como sendo 192.168.1.2. De seguida

procedeu-se à configuração do utilizador *(vpnuser_pptp)*, definindo-se uma palavra passe *(1234)*.

Foi necessário criar uma regra para a passagem do tráfego da *VPN* que assenta em *PPTP*, definindo-se a interface *PPTP VPN*, no protocolo *TCP,* e colocando-se a opção *"any"* para a porta de destino. Para melhor visualização da configuração, consultar *Anexo V – Configuração do PPTP VPN na firewall pfSense*. Procedeu-se à configuração do cliente numa máquina virtual em *Microsoft Windows XP* que corre em cima da plataforma de virtualização da *Oracle*, a *Virtualbox*.

Desta forma procedeu-se à configuração de uma ligação VPN através do *"New Connection Wizard"* do *Microsoft Windows*. A ligação foi configurada com o nome *"ISEP Labors Thesis"*, e tinha como endereço da máquina que fornece a ligação o *192.168.10.252* (*pfSense Firewall*). Nas propriedades da ligação foi necessário definir o tipo de *VPN* que neste caso foi *PPTP VPN*. Por fim, procedeu-se à ligação utilizando o utilizador e a palavra-chave acima descrita, onde se obteve uma ligação por *VPN PPTP* com sucesso. A porta por omissão do *PPTP* é a 1723.

As vantagens da utilização do *PPTP* em *VPN* são:

- *Tunneling* dos protocolos mais comuns IP;
- Permite aos utilizadores a execução de aplicações que usam protocolos específicos, sem que seja necessário alterar a interface do utilizador ou mesmo das aplicações;
- *PPP* multi-ligação, ou seja, caso existam várias ligações físicas, estas podem ser combinadas de forma a aumentar a largura de banda;

Utilização de endereços TCP/IP, mesmo que esses não sejam padrões, o que significa a não utilização de portas conhecidas nas quais normalmente correm serviços conhecidos como por exemplo SMTP (), que por omissão corre na porta 25, ou SSH, que por omissão corre na porta 22, entre outros.

As desvantagens do uso do *PPTP* em *VPN* são:

- O processo de negociação dos parâmetros de ligação é realizado com criptografia muito fraca;
- As mensagens do canal de controlo são transmitidas sem qualquer forma de autenticação ou protecção de integridade;
- Não existe autenticação no período de negociação dos parâmetros da ligação.

Todas as configurações do PPTP VPN podem ser consultadas no Anexo V - Configuração do PPTP VPN na *firewall pfSense* e Anexo VI - Configuração do PPTP em Windows.

8.5.4 Implementação IPSec VPN

O *IPSec* é um conjunto de protocolos que permitem proteger o trafego *IP*, e que utiliza conceitos de autenticação e protecção à integridade, tendo sido desenvolvido pelo grupo de segurança do protocolo *IP* da *IETF*. Por ser um protocolo padrão, existem diversas implementações, como descrito no capítulo 6 desta dissertação.

Nesta implementação de uma *VPN* foi utilizado o *IPSec*, sendo que para isso foi criado na *firewall pfSense* um túnel *IPSec.* De forma a implementar esse túnel foi necessário activar essa funcionalidade. De seguida foi configurada a possibilidade de utilização de clientes bem como a utilização do *IKE*. Foi necessário definir uma *virtual pool* sendo neste trabalho usado o endereço 192.168.1.0/24. Utilizou-se o mecanismo de autenticação *Mutual PSK* em modo agressivo com o identificador a ser o próprio *IP* da máquina cliente. Foi usado o *3DES* como algoritmo de encriptação e o *SHA-1* como algoritmo de sumário.

Usou-se uma chave *Diffie-Hellman* de 1024 *bit*, um tempo de vida de 28800 segundos, e um *NAT* transversal para facilitar o encapsulamento a clientes que residam atrás de uma *firewall* mais restritiva. Estas foram as configurações usadas na fase 1.

Na fase 2 foi configurado o modo de funcionamento em túnel. Foi usado o protocolo *ESP* e o *3DES* como algoritmo de encriptação para uma melhor compatibilidade entre os clientes e a *firewall*. O *SHA-1* foi utilizado como algoritmo de sumário, com um tempo de vida de 3600 segundos, foi definida uma *pre-shared key* e um identificador, utilizando-se o endereço de correio electrónico *(1080003@isep.ipp.pt)* e 1234 como chave.

Por fim, na configuração da *firewall* foram criadas duas regras, uma de tráfego *UDP* nas portas *UDP 500* e *4500*, e outra para tráfego *IPSec*. De seguida foi configurado um cliente numa máquina virtual *Microsoft Windows XP*, sendo para isso utilizado o *software Shrew Soft VPN Access Manager*[1]. No cliente foi definido o *IP*, que neste caso era a *interface WAN* da *firewall* com o *IP* 192.168.10.252, foi usado um *MTU (Maximum Tranport Unit)* 1380 e definido o método de acesso, a utilização de um adaptador virtual.

Ficou definida a utilização da porta 4500, como foi referido anteriormente, e foi criada uma regra para permitir tráfego nesta porta. Nas configurações de autenticação do cliente definiu-se o identificador *(1080003@isep.ipp.pt)*, o *IP* da máquina cliente e por fim a chave partilhada (1234). Nas opções da fase 1 ficou definido o modo agressivo, o *Diffie-Hellman* de 1024 *bit*, e o algoritmo de encriptação, o *3DES*, bem como o *SHA-1* como algoritmo de sumário. Na fase 2 foi escolhida a opção *ESP*, *3DES* e *SHA-1* como algoritmo de sumário, e por fim foi definida a rede remota.

No modo transporte são adicionados cabeçalhos *IPSec* entre o cabeçalho *IP* original e os dados, como exemplificado na Figura 8.5. (Vasques & Schuber, 2002)

[1] Fonte http://www.shrew.net/software

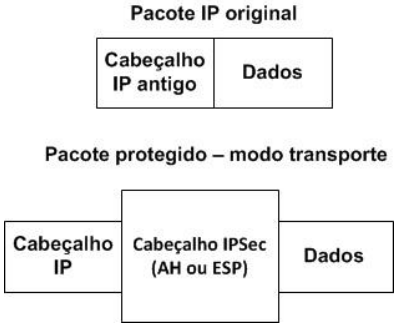

Figura 8.5 - Modo Transporte (adaptada de (Vasques & Schuber, 2002))

O modo transporte é muito utilizado em computadores que estão em redes diferentes, que comunicam entre si e desejam proteger o seu tráfego *IP* por encapsulamento, autenticação ou por ambos. No caso da utilização do modo túnel é também adicionado um cabeçalho *IPSec*, mas a diferença em relação ao modo transporte é que no modo túnel é também adicionado um novo cabeçalho *IP*, sendo que o pacote original é tratado como um dado único. Para isso é encriptado pelo cabeçalho *IPSec* na parte referente ao novo cabeçalho, como se pode ver na Figura 8.6. O modo túnel é utilizado de forma mais comum nas comunicações entre *gateways*, pois fornece uma maior segurança aos dados encriptados no novo pacote. (Vasques & Schuber, 2002)

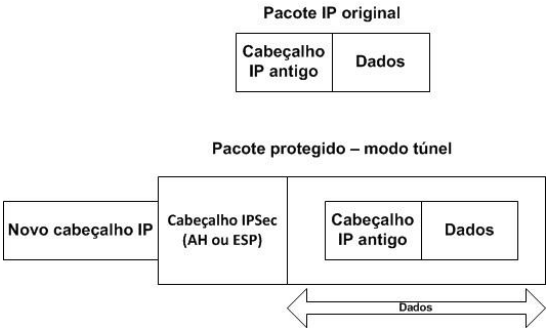

Figura 8.6 - Modo Túnel (adaptada de (Vasques & Schuber, 2002))

As vantagens do uso do *IPSec* são:

- Ser considerado uma norma para tecnologias *VPN*;
- Tecnologia estável;
- Independente de plataforma;
- Aumento de segurança.

As desvantagens da utilização do *IPSec* são:

- Utilização do *IPSec* em sistemas com segurança comprometida;
- Implementação complexa;
- Alto grau de esforço computacional para utilização de algoritmos criptográficos;
- Não protege em caso de ataques do tipo *DoS*;
- Modificações complexas da pilha *IP*;
- São necessários privilégios de administração.

Todas as configurações do IPSec VPN podem ser consultadas no Anexo VII - Configuração do IPSec VPN na *firewall pfSense* e Anexo VIII - Configuração do IPSec em Windows.

8.6 Conclusão

A concepção e a implementação de uma solução para a implementação de uma interligação de forma segura de uma rede de laboratórios remotos distribuídos geograficamente, envolve diversos conceitos informáticos tais como: redes; segurança; laboratórios remotos; redes privadas; criptografia entre outros. As três soluções estudadas e implementadas neste trabalho procuraram ser uma resposta à questão de como garantir segurança neste tipo de redes.

Assim sendo, e após a implementação das três soluções, a que melhor se adequa a este trabalho foi a solução implementada em OpenVPN, pois permite uma ligação servidor-"multi-cliente", que funciona como concentrador das diversas redes em que são utilizadas as experiências remotas distribuídas geograficamente. Esta solução permite uma compatibilidade com a maioria dos sistemas operativos. Como todas as soluções, também apresenta as suas desvantagens, que neste caso é o facto de não ser baseada em nenhum cliente Web e também de não se integrar com nenhuma outra solução VPN.

Na

Tabela 2 pode observar-se as vantagens da utilização desta tecnologia em relação às demais implementadas.

Tabela 2 - Comparação de soluções implementadas no trabalho

	OpenVPN	IPSec	PPTP
Custos de ligação	(+)	(-)	(-)
Custos de Administração	(+)	(-)	(-)
Escalabilidade	(+)	(-)	(-)
Compatibilidade com Sistemas Operativos	(=)	(=)	(-)
Configuração e implementação	(+)	(-)	(-)
Criptografia	(=)	(=)	(-)

Legenda: (+) Vantagem; (=) Igualdade; (-) Desvantagem.

9 Conclusão

Neste capítulo é realizado um resumo de todo o trabalho elaborado ao longo deste estudo e que foi descrito nos capítulos anteriores. Nos pontos seguintes será efectuada uma comparação entre os objectivos propostos e os resultados alcançados.

9.1 Objectivos Alcançados

Este trabalho teve como objectivo o estudo da segurança em redes e a implementação de uma solução de comunicação segura para laboratórios remotos distribuídos geograficamente. De forma a minimizar custos, toda a implementação foi assente em *software* de código aberto e na utilização de um computador de baixo custo.

No que respeita à criação das *VPNs*, estas foram configuradas de modo a permitir obter os resultados pretendidos na criação de uma ligação segura para laboratórios remotos. As tecnologias *VPN* usadas na obtenção dos resultados foram o *OpenVPN, PPTP* e *IPSec*. Desta forma garantiu-se o uso de tecnologias que permitem minimizar os custos da implementação de *VPNs*, não sendo necessário recorrer a soluções proprietárias que tornem a solução dispendiosa.

O *pfSense* mostrou-se a escolha acertada visto que suporta nativamente quaisquer tecnologias que foram estudadas e implementadas, sem necessidade de usar recursos físicos muito caros, permitindo o uso de tecnologias de código aberto sem comprometer a segurança no funcionamento das soluções que suportam a segurança nas comunicações dos laboratórios remotos.

9.2 Trabalho Futuro

Com vista ao aproveitamento do que foi desenvolvido neste trabalho e com o intuito de continuar a sua evolução seria importante tentar melhorar as regras de segurança da *firewall*, de forma a criar regras mais restritivas que possibilitem uma maior segurança na solução apresentada. No caso deste trabalho optou-se pela utilização de portas por omissão nas regras apresentadas pois seria apenas para utilização em modo experimental, assim desta forma a solução fica mais vulnerável, para contornar este problema é aconselhado a utilização de portas dinâmicas. A arquitectura da solução permite que as alterações possam ser realizadas sem qualquer custo adicional e permite a reutilização de *hardware* antigo, contribuindo desta forma para um menor gasto em termos de *hardware*.

O desenvolvimento deste trabalho poderá levar a uma implementação aplicada ao *OPEN WRT*, que é uma distribuição em *Linux* que está disponível para cliente embutidos. (openwrt.org, 2011) Como mais valia poderá aproveitar-se o facto do *OPEN WRT* não ser estático e permitir a configuração de novos pacotes que permitem alcançar os objectivos desejados, que neste caso se centram em proporcionar segurança nas comunicações dos laboratórios remotos geograficamente distribuídos. Com a aplicação desta solução em *OPEN WRT* tornar-se-á possível criar uma configuração personalizada. Desta forma pode funcionar numa arquitectura tipo *"pc embutido"* e assim obter um baixo consumo energético.

No que respeita a trabalho futuro na área de clientes embutidos, poderá usar-se este tipo de clientes de forma a criar um *hardware* que seja capaz de integrar qualquer uma das soluções *VPN* implementadas, funcionando como um mecanismo automático de ligação de experiências que estejam geograficamente distribuídas.

9.3 Considerações Finais

Em síntese, após avaliar todas as configurações e considerações teóricas desenvolvidas ao longo dos capítulos desta dissertação, é possível afirmar que este trabalho oferece uma solução de baixo custo na criação de redes *VPN* que permite segurança na implementação de laboratórios remotos distribuídos.

Ao longo da dissertação foram apresentados os vários passos necessários à criação e configuração das soluções bem como as diferenças entre elas. Este trabalho permitiu a aquisição de competências nas áreas de redes *VPN*, conceitos de segurança em redes e tecnologias de código aberto, pretendendo ser desta forma um guia na implementação deste tipo de soluções.

A utilização do *pfSense* foi uma experiência que permitiu adquirir conhecimentos que serão úteis no desenvolvimento de actividades relacionadas com a vertente profissional, pois tornou-se possível compreender o funcionamento da ferramenta e utilizar algumas das potencialidades desta poderosa solução. Foi também possível adquirir conhecimentos práticos no que respeita à implementação de *VPN* e renovação dos conceitos teóricos no que respeita à temática da segurança.

10 Referências

Aqun, Z., Yuan, Y., Yi, J., & Guanqun, G. (2000). *Research on Tunneling Techniques in Virtual Private Networks.* Communication Technology Proceedings, 2000. WCC - ICCT 2000. International Conference on , vol.1, pp.691-697 vol.1, 2000. Obtido em 16 de Maio de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=889294

Badra, M., & Urien, P. (2008). *TLS Tandem*. New Technologies, Mobility and Security, 2008. NTMS '08. , pp.1-5, 5-7 Nov. 2008. Obtido em 06 de Maio de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4689153

Berger, T. (2006). *Analysis of current VPN technologies*. Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on , pp. 8 pp., 20-22 April 2006 . Obtido em 07 de Maio de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1625300

Casad, J. (2001). *Sams Teach Yourself TCP/IP in 24 Hours* (2nd ed. ed.). Sams.

Chou, W. (2002). *Inside SSL: the secure sockets layer protocol.* IT Professional , vol.4, no.4, pp. 47- 52, Jul/Aug 2002. Obtido em 06 de Maio de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1046644

Coron, J.-S. (2006). *What is cryptography?* Security & Privacy, IEEE , vol.4, no.1, pp. 70- 73, Jan.-Feb. 2006. Obtido em 29 de 04 de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1588831

de Borde, D. (2008). *http://www.insight.co.uk/files/whitepapers/Two-factor%20authentication%20(White%20paper).pdf.* Obtido em 22 de 04 de 2010

de Canniere, C., Biryukov, A., & Preneel, B. (2006). *An introduction to Block Cipher Cryptanalysis.* Proceedings of the IEEE , vol.94, no.2, pp.346-356, Feb. 2006. Obtido em 29 de Abril de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1580504

de Rezende, E. R. (2004). *Segurança no Acesso Remoto VPN*. Obtido em 14 de Maio de 2012, de http://www.las.ic.unicamp.br/paulo/teses/20040227-MSc-Edmar.Roberto.Santana.de.Rezende-Seguranca.no.acesso.remoto.VPN.pdf

Dhall, H., Dhall, D., Batra, S., & Rani, P. (2012). *Implementation of IPSec Protocol*. Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference on , pp.176-181, 7-8 Jan. 2012. Obtido em 08 de Maio de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6168355

Hussain, M., & Seret, D. (2006). *Extending TLS for Trust Delegation in Home Networks.* Advanced Communication Technology, 2006. ICACT 2006. The 8th International

Conference , vol.1, pp.497-502, 20-22 Feb. 2006. Obtido em 06 de Maio de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1625621

IEEE, 2010. *802.1X-2010 - IEEE Standard for Local and metropolitan area networks - Port-Based Network Access Control.* Obtido em 01 de Agosto de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5409813

IEEE, 2012. *IEEE 802.3 Ethernet Working Group*. Obtido em 21 de Julho de 2012, de http://www.ieee802.org/3/

Junior, J. (2007). *Laboratórios Baseados na Internet: desenvolvimento de um laboratório virtual de química na plataforma MOODLE.* Obtido em 6 de Janeiro de 2011, de http://nautilus.fis.uc.pt/cec/teses/joaojunior/docs/tesecompleta.pdf

Kasper, M. (2011). *m0n0wall*. Obtido em 30 de Julho de 2012, de http://m0n0.ch/wall/

Kehe, W., & He. Jianping, D. (2011). *Secure wireless remote access platform in power utilities based on SSL VPN.* Information Technology and Artificial Intelligence Conference (ITAIC), 2011 6th IEEE Joint International , vol.1, pp.93-97, 20-22 Aug. 2011. Obtido em 06 de Maio de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6030159

Kuihe, Y., & Xin, C. (2007). *Implementation of Improved VPN Based on SSL.* Electronic Measurement and Instruments, 2007. ICEMI '07. 8th International Conference on , pp.2-15-2-19, Aug. 16 2007-July 18 2007. Obtido em 06 de Maio de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4350641

Kurose, J., & Ross, K. (2004). *Computer Networking: A Top-Down Approach Featuring the Internet* (Third Edition ed.). Addison Wesley.

Li, Y., Zhao, G., Liu, T., & Du, L. (2009). *The research on Cryptographical System of IPSEC VPN based on Combined Symmetric Key*. Network Infrastructure and Digital Content, 2009. IC-NIDC 2009. IEEE International Conference on , pp.999-1003, 6-8 Nov. 2009. Obtido em 28 de Dezembro de 2011, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5360958

Loshin, P. (2004). *TCP/IP Clearly Explained* (Fourth Edition ed.). Morgan Kaufmann.

Magalhães, H., & Grilo, A. (2006). *A SEGURANÇA INFORMÁTICA E O NEGÓCIO ELECTRÓNICO.* Porto: SPI – Sociedade Portuguesa de Inovação Consultadoria Empresarial e Fomento da Inovação, S.A.

Martins, D. (2000). *Redes Privadas Virtuais com IPSec*. Obtido em 07 de Maio de 2012, de http://www.cic.unb.br/docentes/pedro/trabs/vpn.pdf

Matias, P. (2007). *TUN/TAP - Redes virtuais e suas aplicações praticas*. Obtido em 30 de Junho de 2012, de http://www.thebugmagazine.org/magazine/bug02/0x04_tun-tap.txt

Mendyk-Krajewska, T., & Mazur, Z. (2010).. *Problem of network security threats.* Human System Interactions (HSI), 2010 3rd Conference on , pp.436-443, 13-15 May 2010. Obtido em 22 de 04 de 2012, de IEEE: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5514533

Menezes, A., van Oorschot, P., & Vanstone, S. (1996). *Handbook of Applied Cryptography, 1st ed.* CRC Press.

Mostafa, M.-S., Deif, S., & Kholidy, H. (2008). *ULTRA GRIDSEC: Peer-to-Peer Computational Grid Middleware Security Using High Performance Symmetric Key Cryptography*. Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on , pp.137-142, 7-9 April 2008. Obtido em 29 de Abril de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4492468

Narayan, S., Kolahi, S., Brooking, K., & Vere, S. (2008). *Performance Evaluation of Virtual Private Network Protocols in Windows 2003 Environment*. Advanced Computer Theory and Engineering, 2008. ICACTE '08. International Conference on pp.69-73, 20-22 Dec. 2008. Obtido em 07 de Maio de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4736924

Nedic, Z., Machotka, J., & Nafalski, A. (2003). *Remote Laboratories versus Virtual and Real Laboratories*. Obtido em 15 de Janeiro de 2011, de http://fie-conference.org/fie2003/papers/1077.pdf

openwrt.org. (2011). *openwrt.org*. Obtido em 31 de Julho de 2012, de openwrt.org: https://openwrt.org/

Plummer , D. C., 1982. *RFC 826*. Obtido em 21 de Julho de 2012, de http://www.ietf.org/rfc/rfc826.txt

Portmann, M., & Seneviratne, A. (2001). *Selective security for TLS*. Networks, 2001. Proceedings. Ninth IEEE International Conference on , pp. 216- 221, 10-12 Oct. 2001. Obtido em 07 de Maio de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=962343

Romney, G., & Parry, D. (2006). *A Digital Signature Signing Engine to Protect the Integrity of Digital Assets*. Information Technology Based Higher Education and Training, 2006. ITHET '06. 7th International Conference on , pp.800-805, 10-13 July 2006. Obtido em 29 de Abril de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4141711

Sakalli, M., Bulus, E., & Buyuksaracoglu, F. (2004). *Cryptography education for students.* nformation Technology Based Higher Education and Training, 2004. ITHET 2004. Proceedings of the FIfth International Conference on , pp. 621- 626, 31 May-2 June 2004. Obtido em 29 de Abril de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1358246

Santos, R., Bauchspiess, A., & Borges, G. (2004). *Laboratório Remoto de Automação Predial*. Obtido em 23 de Janeiro de 2011, de http://lara.unb.br/~gaborges/arquivos/pub.cobenge2004.santos.pdf

Sicero, J. (2004). *Pilha TCP/IP Adaptável à Aplicação*. Obtido em 20 de Janeiro de 2012, de http://www.lisha.ufsc.br/pub/Sincero_BSC_2003.pdf

Srinivas, B., & Chan, T. (2003). *Security threats and risks with content transformation intermediaries.* Telecommunications, 2003. ICT 2003. 10th International Conference on , vol.1, pp. 608- 613, 23 Feb.-1 March 2003. Obtido em 17 de 04 de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1191479

Taherdoost, H., Zamani, M., & Namayandeh, M. (2009). *Study of smart card technology and probe user awareness about it: A case study of Middle Eastern students.* Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on , pp.334-338, 8-11 Aug. 2009. Obtido em 27 de Julho de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5234410

Teixeira, L., Costa, O., Pereira, V., Leão, C.P. , Soares, F.O., Restivo, M.T., Chouzal, M.F., Mendes, J. & Campos, J. C. (2005). *Laboratórios virtuais: duas aplicações no ensino de engenharia*. VII Simpósio Internacional de Informática Educativa , 479-481, Editado por A. Mendes, I. Pereira and R. Costa, Leiria, Portugal, 2005.

Titz, O. (2001). *Why TCP Over TCP Is A Bad Idea*. Obtido em 09 de Julho de 2012, de http://sites.inka.de/bigred/devel/tcp-tcp.html

Vaithiyanathan, Gracelin, S., Edna, E., & Radha, S. (2010). *A Novel Method for Detection and Elimination of Modification Attack and TTL Attack in NTP Based Routing Algorithm.* Recent Trends in Information, Telecommunication and Computing (ITC), 2010 International Conference on , pp.60-64, 12-13 March 2010. Obtido em 17 de Abril de 2012, de http://ieeexplore.ieee.org/ielx5/5459230/5460529/05460612.pdf?tp=&arnumber=5460612&isnumber=5460529

Vasques, A., & Schuber, R. (2002). *Implementação de uma VPN em Linux utilizando o protocolo IPSec*. Obtido em 30 de Junho de 2012, de http://www.abusar.org.br/manuais/VPN-alan-rafael.pdf

Wu, J. (2009). *Implementation of Virtual Private Network based on IPSec Protocol*. Future Computer and Communication, 2009. FCC '09. International Conference on , pp.138-141, 6-7 June 2009. Obtido em 16 de Maio de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5235687

Yang, B., Li, Z., Zheng, S., & Yang, Y. (2009). *Hash function construction based on coupled map lattice for communication security*. Global Mobile Congress 2009 , pp.1-7, 12-14 Oct.

2009. Obtido em 29 de Abril de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5295892

Yasinsac, A., & Childs, J. (2001). *Analyzing Internet security protocols*. High Assurance Systems Engineering, 2001. Sixth IEEE International Symposium on , pp.149-159, 2001. Obtido em 07 de Maio de 2012, de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=966816

11 Anexos

(no CD-ROM)

Anexo I - Instalação da *firewall pfSense*

Anexo II - Configuração inicial da *firewall pfSense*

Anexo III - Configuração do *OpenVPN* na *firewall pfSense*

Anexo IV - Configuração do *OpenVPN* no *Windows* e *Linux*

Anexo V - Configuração do *PPTP VPN* na *firewall pfSense*

Anexo VI - Configuração do *PPTP* no *Windows*

Anexo VII - Configuração do *IPSec VPN* na *firewall pfSense*

Anexo VIII - Configuração do *IPSec* no *Windows*