Beyond Samba - Integrating Windows Authentication into a wireless VPN solution

Andrew Bartlett

4th January 2005

Contents

List of Figures

Chapter 1

Introduction

This document describes the efforts by a project team at the Department of Computer Science (ANU), to provide secure access to its internal networks, by means of a 802.11b wireless network. It describes the functional system, as well as the software and technology used to construct it.

1.1 The wireless network

The wireless network on which this solution was built is ANUNorth[1]. Built on the 802.11b[20] wireless Ethernet (WiFi[21]) standard, it services a number of departments in the northern part of ANU's campus. This network has been built from commodity Wireless Access Points (WAP), and linked together on a single Virtual Local Area Network (VLAN).

1.1.1 The problem

Wireless access presents an inherit risk to networks, be it corporate, government, personal or academic. This is because (and often for the first time) the network extends outside the building - removing the usual requirement for at least physical access to a network port.

A wireless network poses a very particular problem for security, simply because an attacker cannot even be traced to a network port - only to the region of wireless network coverage. From this distance, risks range from the provision of potentially expensive Internet access, to theft of data (data carried across a radio network is easily intercepted) and untraceable system compromise.

[1]So named because it spans a number of buildings in the northern quadrant of the Australian National University.

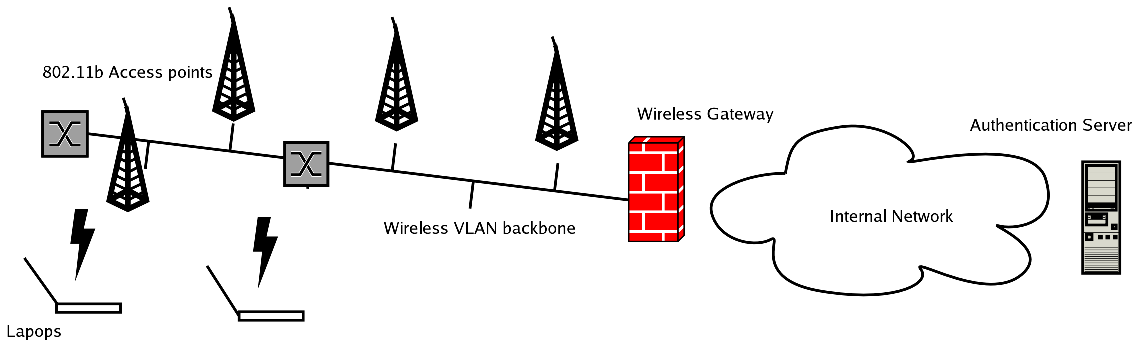


Figure 1.1: Wireless network overview

Despite this, users still demand simple ubiquitous access to their data and the Internet, without having to install extra software or perform extensive configuration.

1.1.2 The Shared Network

A futher problem that any solution needed to solve is that of the shared ANUNorth network. It was a requirement that multiple seperate bodies within the university could provide access to their own clients, without conflicting with each other.

1.1.3 Possible solutions

Wired Equivalent Privacy

The most traditional method used to secure a wireless network is the 'Wired Equivalent Privacy' (WEP) provided in the 802.11b standard. Available in 64 and 128 bit strengths, this method has been discredited cryptographically, and shown to be infeasible to secure in any case. WEP's weakness[2] comes from the need to share a common key between all participating hosts. WEP's strength is that it is easily configured on almost all clients and access points - simply input a 'password' or key into the appropriate configuration dialogue.

MAC Filtering

Another method traditionally used to protect networks from intrusion (but not sniffing) involves a simple filter based on the MAC address (the 'hardware address' of an individual network device). On small networks, this can be quite effective, when combined with WEP. Such a network may present sufficient deterrent to avoid the interest of a casual attacker, and it is easily configured. Unfortunately, MAC addresses can easily be spoofed (as the address of each card must be sent in the clear, and is user-configurable). Also, on a larger network, these access controls must be entered into each and every WAP, which may become tedious.

VPN technology

The solution chosen for the application was the use of an IP-layer Virtual Private Network (VPN). Virtual Private Networks provide a way to both secure and authenticate sensitive data traveling between a distant node on an untrusted network and the trusted internal network. In the case of the wireless LAN, that untrusted network is the wireless network itself. By wrapping all that data into a cryptographically secure stream, the wireless link layer can remain as commodity hardware, unaffected by the stronger authentication system.

The use of a VPN also allows an arbitary number of independent VPN gateways, each simply operating on it's own IP address.

Wi-Fi Protected Access / 802.1x

There are emerging standards for wireless network protection, based on wireless access point level security, and some form of password authentication. The major issue with these systems is that they will require the replacement of every access point to enable the new technology. Similarly, all clients will need to updated with compatible client software.

1.1.4 VPN solutions

There are a large number of potential VPN solutions available, including:

CIPE[5] Cryptographic IP encapsulation. Wraps IP into UDP (User Datagram Protocol) packets, and encrypts them. Key-based. Manual configuration.

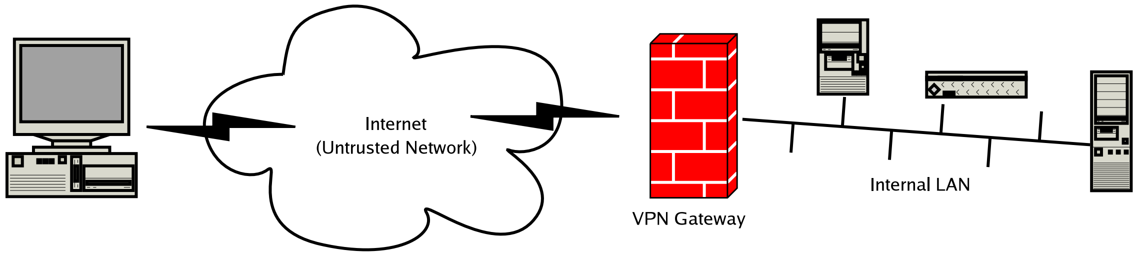


Figure 1.2: Classic VPN architecture

IPSec[4]	IPSec is security and encryption layer for IP (the Internet Protocol).
ppp-over-ssh	A 'hack' VPN solution, which uses SSH public keys to authenticate a stream of data in PPP (Point to Point Protocol) format. Uses normal SSH on TCP port 22.
PPTP[6]	Microsoft's VPN technology - wraps PPP inside GRE (Generic Routing Encapsulation) packets, for transmission to the VPN server. Uses passwords for both authentication and for the basis of the encrypted stream. Dynamic configuration.
Proprietary	A number of vendor-specific proprietary VPN solutions exist, all routing over IP in some way, and using all manner of authentication technologies.

The choice of VPN technology is strongly influenced by the clients expected to be using the network, and the per-client administration overhead of the proposed solution.

Only the Microsoft PPTP solution can be implemented on both MacOS X and Microsoft Windows 98/2000/XP without client modifications. This strongly influenced the decision to use it, as this will allow the vast majority of users to 'help themselves' in setting up their clients. In particular, this avoids the need to install potentially incompatible software on MacOS X, Win98, Win2000 and WinXP client devices.

The PPTP system is also available on the Linux platform, which must be supported in any eventual solution. Because PPTP is available for Linux servers, this solution was chosen for the student wireless access solution.

1.1.5 Standards, RFCs and nomenclature

This document will refer to a number of networking concepts, particularly those associated with use of the Internet Protocol (IP), Point to Point Protocol (PPP) and an Ethernet Local Area Network (LAN). The documents referred to as RFCXXXX are Internet 'Request for Comments' documents. Some of these documents are ratified standards of the Internet Engineering Task Force (IETF), a standards body, while others are more informal documents that are published to explain an existing internet practice. All are readily available on-line, and are uniquely identified by RFC number.

Chapter 2

Point to Point Protocol

2.1 Introduction

Microsoft's VPN technology of choice is PPTP (Point To Point Tunneling Protocol). A simple wrapping of PPP over IP, it provides no inherent security benefit on its own. Indeed, for the purposes of this exercise, PPPoE wrapping (Point to Point Protocol over Ethernet) or any other host-to-host PPP connection would have been quite sufficient. What makes PPTP useful however is the PPP options that are traditionally associated with its use (those being 'secure' authentication and data encryption).

2.2 PPP

The basis for the VPN solution is PPP, the Point to Point Protocol described in RFC1661. PPP provides a generic way to encapsulate a number of network protocols, but particularly IP datagrams, over a point-to-point link. For IP transfer, each end of this virtual circuit establishes an IP address, and packets may be routed over this link to other networks. This link may optionally be authenticated, and a data encryption scheme has been devised.

2.2.1 PPP encapsulation

PPP is often used over serial links, in dial-up Internet access. Indeed, this is where most people first meet the protocol. However, a number of standards have been implemented whereby this protocol can be carried over another:

PPPoE	Point to Point Protocol over Ethernet, described in RFC 2516.
PPTP	Point to Point Tunneling Protocol, described in RFC 2637

PPTP provides a way to create a (possibly authenticated, possibly encrypted) data stream over IP networks, but does not modify the PPP layer, or it's semantics. This made it very attractive for use as a VPN protocol - particularly as PPP semantics are well-known, and the protocol is extensible. PPTP uses standard TCP/IP for its control protocol, and GRE (Generic Routing Encapsulation) for the main data stream.

2.2.2 PPP Authentication

PPP authentication has traditionally been by one of the following:

PAP	Password Authentication Protocol, described in RFC 1334.

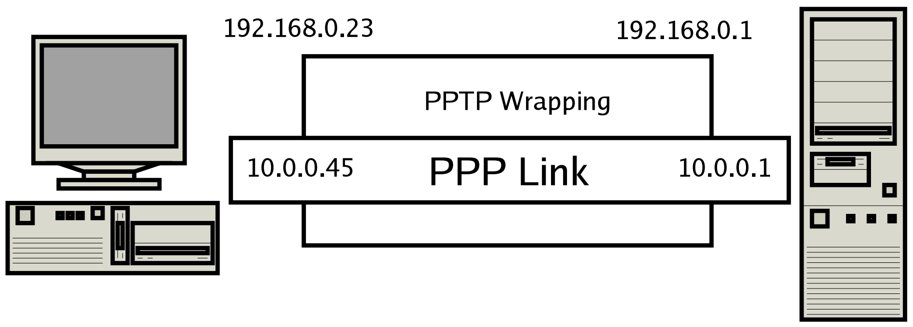


Figure 2.1: PPTP is a wrapping around standard PPP

CHAP	Challenge Handshake Authentication Protocol, described in RFC 1994.

PAP provides a very simple authentication system - and a system that is compatible with almost any authentication back-end - as once the plain-text password is received by the server, it may be processed into any form the system requires. CHAP provides a secure way to prove knowledge of a shared password, between the PPP client and server. its advantage over PAP is that the plain-text password need never travel over the link. The primary disadvantage of CHAP is that it cannot easily be integrated into an existing authentication database.

When designing an authentication system for VPN use, Microsoft decided to implement its own PPP Authentication layer - closely mirroring the NTLM authentication system used inside its native networks. This protocol was called MSCHAP, and is described in RFC 2433, and is essentially the existing NTLM authentication in a slightly different wrapping. This allowed the use of existing authentication databases and infrastructure, while still obscuring the password as it traveled over the network.

> Microsoft created MSCHAP to authenticate remote Windows workstations, providing the functionality to which LAN-based users are accustomed while integrating the encryption and hashing algorithms used on Windows networks. (RFC 2433).

A number of security issues were found with this original implementation, and were addressed with MSCHAPv2 (RFC 2759) - a discussion of the design, strengths and weaknesses of this protocol follows in later chapters.

2.2.3 PPP Encryption (MPPE)

PPP has no native method for the encryption of data carried over the network. Indeed, this task is often carried out either above or below the PPP layer. However, in the Microsoft VPN implementation, PPP's compression layer has been adjusted to become an encryption layer. Given encryption keys from the authentication process, a scheme known as MPPE (Microsoft Point to Point Encryption)[12], described in RFC 3078 is used to encrypt the data. Essentially, this is RC4[17, 18] (a stream cipher) applied to the PPP packets. This is what provides the 'private' part of Microsoft-compatible Virtual Private Networks.

2.3 Linking it all together

Each of these layers on their own does not provide any particular value in constructing a VPN solution, nor a solution to the original problem posed in the introduction. Firstly, the PPTP layer allows the linking of arbitrary IP-connected computers for a PPP session. When configured for VPN use, only the MSCHAP or MSCHAPv2 (depending on client/server software versions) authentication will be allowed. Each of these authentication protocols produces, as a side-effect, a ‘session key’ - a password-derived value that is known to both the client and the server. This session key is used to setup MPPE encryption on the Internet link. For VPN use, MPPE is mandatory.

While PPPoE could certainly play the same role in our architecture, this would not be supported ‘out of the box’ on Microsoft Windows clients - this is a key requirement to reduce support costs.

Chapter 3

MSCHAP

The role of MSCHAP in the VPN solution is to provide an authentication protocol which is compatible with existing Microsoft Windows networking environments. While such integration has significant drawbacks, the design choice here was to allow 'single sign on.' The use of MSCHAP allowed Microsoft to implement that, while retaining VPNs as an 'edge service,' i.e. one that required no changes to central infrastructure. This consideration can be clearly seen in the design and implementation of MSCHAP and MSCHAPv2.

3.1 MSCHAP design principles

MSCHAP is designed for operation on what is termed a 'domain member server;' that is, a server that has established itself as a trusted member of a Microsoft authentication domain. A domain is a collection of computers, sharing access to a common, central password database. Access to any of the members of the domain is by the same, centrally maintained user-name and password. Each network connection involves Microsoft's particular brand of 'challenge-response' authentication, and the authentication is often transparent to the user.

3.1.1 Challenge-Response Authentication

Challenge-response authentication avoids the need to transfer the plain-text password over the network, by use of one-way functions. These functions, known as 'digest' or 'hash' functions are considered mathematically impossible to reverse, and as such do not reveal their inputs in the final result. Random Data (often called a challenge, or more formally a *nonce*) is introduced to ensure that subsequent authentications will always produced a different output, known as the response.

Each side of the authentication exchange performs that same mathematical process. The server determines that the client is authenticated by simply comparing the result it found with the result the client provided. Traditionally, the server (and possibly the client) provides some random data to 'seed' this exchange, ensuring that past results cannot be used in a future exchange. This way, the plain-text password (nor its equivalent) never passes over the network.

Session Key Generation

Because each party to the authentication exchange must have a copy of the user's password, it is possible to use this shared secret to encrypt data. In Microsoft networking, this is termed the 'user session key.' In MPPE (the encryption method used with MSCHAP) a derivative of the user session key is used to encrypt to contents of the VPN.

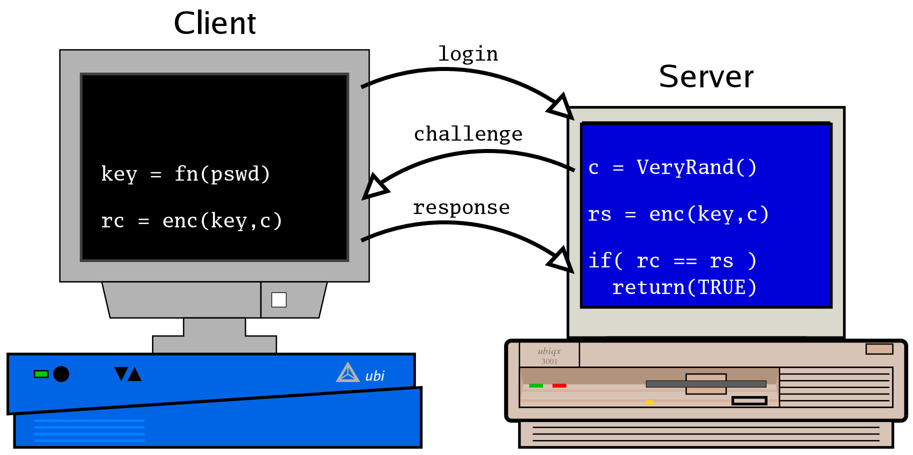


Figure 3.1: Challenge/Response

The server generates a random challenge, which it sends to the client. Both systems encrypt the challenge using the secret encryption key. The client sends its result (rc) to the server. If the client's result matches the server's result (rs), then the two nodes have matching keys.[1]

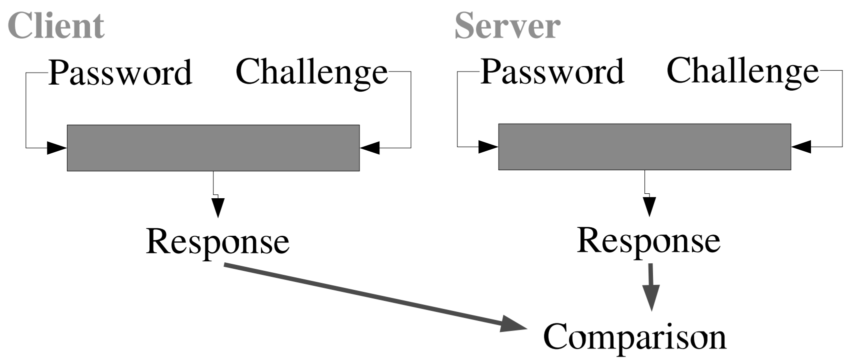


Figure 3.2: Use of One-Way-Functions in Challenge-Response Authentication

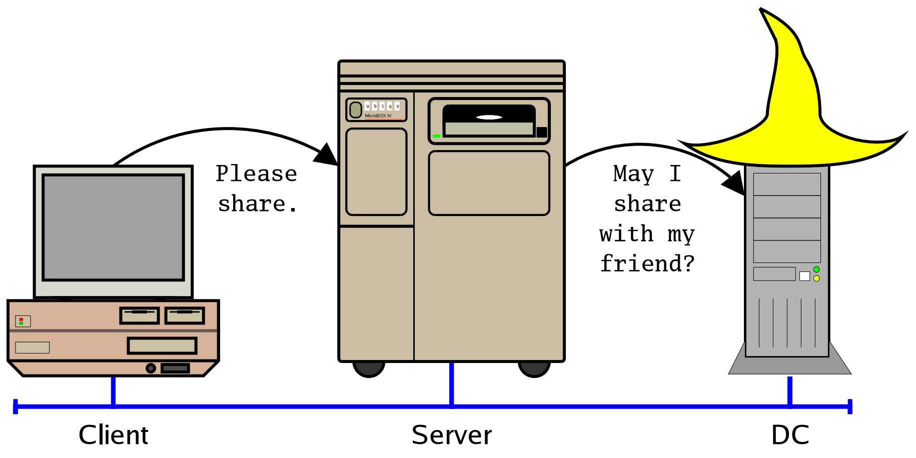


Figure 3.3: Centralized Authentication.

> The Domain Controller (DC) wears a special hat. It keeps track of the common authentication database that is shared by the SMB servers in the Domain. The SMB servers query the DC when a client requests access to SMB services. 2

In the same way, the VPN server asks 'may I let my client in,' when a VPN client comes knocking at its door. Microsoft went out of its way to make sure these two processes looked the same.

3.1.2 Edge server abstractions

One of the most important considerations in the Microsoft VPN design (and an aspect that we are also applying in our implementation of this design) is the separation of the VPN server from the password store. For simple security reasons this is a good thing, but more importantly, this allowed the deployment into existing, otherwise untouched NT Domains. The system for abstracting this authentication is the same one that is used for file-share connections internally.

NT Domains

NT Domains are the core of Microsoft's networking authentication world. A server, combining with a number of synchronized peers, holds a list of all the user-names and passwords for a given 'domain.' A domain is essentially a collection of computers that agree to use this common database for authentication. Even with the move to 'Active Directory,' this concept of a shared authentication database, and the protocols used are still very much the same.

VPN servers in the Domain

The VPN server acts as a 'man in the middle' between the VPN client and the central machine handling the authentication database the Domain Controller (DC). In this configuration the VPN server generates the challenge, supplies it to the client, which constructs a response. Both the challenge and the response are securely communicated to the DC, which applies the same mathematical test described above. The DC provides a 'yes or no' answer to the VPN server, but for success it also supplies a secret value derived from the password called the 'user session key,' which may be used to encrypt communications. Even without the original password, the VPN server may 'securely' encrypt the tunnel.

3.2 MSCHAP Design Flaws

MSCHAP[10] (v1) provides a very simple reworking of Microsoft's own challenge-response authentication system, known as NTLM authentication. After MSCHAP was ripped apart[7], work was done to provide a more secure system, but on the same basis. Microsoft's 'new and improved' authentication scheme[13, 11] still inter-operating as an 'edge server' on a windows network, without requiring changes to the NT Domain Controller.

3.2.1 The original problem

The base problem with MSCHAP (version 1) is the NTLM authentication scheme. Already showing its age and design flaws[9] on the trusted internal LAN, NTLM was far from suitable for deployment onto the Internet. Fundamentally, the flaws in MSCHAP are the use of LANMAN (or LM) password hash, and the LANMAN challenge-response system - a system confined to uppercase, ASCII passwords. To add insult to injury, this weaker password was normally accompanied by a much stronger 'NT' password. The server would check the stronger of the two, but nothing stopped the attacker breaking the weaker one.

The other design flaw in the original MSCHAP system was the 'session key' used for encrypting the data stream. Based only on the user's password, it did not change between sessions, and was the same in each direction. Trivial manipulation of the data in the 'encrypted' stream could be used to reveal this secret key, which would not change between sessions[7]!

Finally, there is no verification that the user is connected to a 'trusted' server, except by the successful decryption of the tunneled datagrams.

Had it not been the 'established' cryptosystem inside Microsoft Windows networks, this authentication scheme would hopefully have never have been established, and certainly should never have been used for Internet or VPN authentication.

3.2.2 A compatible solution

The need for a newer version of MSCHAP is obvious. It is less obvious is how to construct such a system, while ensuring that only the VPN server itself would require such an upgrade. The result was MSCHAPv2. MSCHAPv2 still uses the NTLM authentication scheme, although, fortunately, it has dropped the LM authentication. This scheme focuses on avoiding what are termed 'chosen plain-text' attacks on the protocol. By including random data of the client's choosing (the client challenge), a malicious server cannot force a particular cryptographic outcome. All this is forced into the 8-byte 'challenge,' previously provided by the server alone. Likewise, both the client and the server contribute random data into the generation of the 'session key' used to encrypt the data stream. This ensures that session key keeps changing.

Even with all these differences, the same NTLM algorithm is applied, allowing existing domain controllers can remain unaware of the changes. An unmodified domain controller supplies the original session key to the VPN server. The VPN server then manipulates that 'long term' session key for use on the VPN connection, by introducing random data into it.

Unfortunately compatibility comes at a high price, and so MSCHAPv2 also retains the maximum possible encryption strength at 56bit, due to the way the DES function is still applied in the challenge-response step.

3.2.3 A better solution

A better solution to this problem would have involved starting from scratch - a new, secure authentication scheme (like NTLMv2 or, better still, a scheme resilient to offline dictionary attacks) should have been used, and a patch provided for the Domain Controllers involved. This would have allowed for 'true' 128-bit encryption of the VPN link, and would have been much more suitable in an Internet environment.

3.3 MSCHAPv2 in detail

It is assumed that the reader has read and understood the references on NTLM Authentication - particularly Chapter 2.8 of *Implementing CIFS*[1], as that information will not be covered again. As has been mentioned, MSCHAP is nothing more than NTLM authentication, in a PPP-friendly wrapping. The exact format is described in RFC 2433, and the algorithms themselves are well-understood. Chapter 15 of *Implementing CIFS* describes this in detail, in the context of CIFS file-sharing. The MSCHAP implementation is no different in PPP, only slightly renamed and in the PPP packet.

MSCHAPv2 is a slightly different beast, designed to overcome server-side and man-in-the-middle attacks on the protocol. It does this by introducing a client-specified *nonce* into the calculations, and by returning a *server verifier* to prove that the server really does know the password (and isn't just saying 'yes' to all incoming connections). It does this by replacing the server-supplied nonce (the server challenge) with a combination of client and server inputs. This way, neither the client, nor the server, can independently force a particular mathematical result.

3.3.1 The challenge-response algorithm

The inputs to the challenge-response algorithm are:

- 16 bytes of server-supplied nonce
- 16 bytes of client-supplied nonce
- The user-name, as supplied by the client
- The password (as an NT password hash)

The SHA1 algorithm

When implementing the one way function components of MSCHAPv2, the Secure Hash Standard (SHA1)[16] was the chosen algorithm. This hash function generates a 192 bit hash of the data, and is highly resistant to both reversal and collisions. Its use seems overkill (the DES function applied to other stages is 56bit at best), and in other NTLM derivations the more common MD5 digest function is used.

Generating the Challenge Hash

LM challenge-response algorithm requires 8 bytes of input, as a challenge. This is obtained by means of the SHA1 digest function.

The challenge is specified by:

```
    challenge = head(sha1(concat(server_nonce, client_nonce, username)), 8)
```

That is, the first 8 bytes of the SHA1 function over the server nonce, the client nonce and the user-name. This is used as an input into the LM challenge response function.

Generating the NT password

MSCHAPv2 uses the NT password hash as the input into the LM challenge-response algorithm. It uses the MD4[15] digest function over the little-endian two byte Unicode (UCS2-LE)[14] representation of the user's password for this value. (This results in a 16 byte value.)

```
    nthash = MD4(unicode(password))
```

Generating the Challenge Response

The LM challenge-response algorithm is used for MSCHAPv2 as such:

```
    NTLMresponse = LM_response(nthash, challenge)
```

This is compatible with the implementation used in standard NTLM, as the challenge is still 8 bytes, and still used in the same way. This allows the use of unmodified NT domain controllers.

3.3.2 The NT Session Key

The NT Session key is a fixed, secret value associated with each NT password. It is the result of MD4() applied over the nt password hash, and is therefore as secret as the NT password hash, but without being the password itself.

The NT session key is important because it is the only 'secret' value that is disclosed back to the VPN server, by an NT domain controller. It is used in the Authenticator response, and in the encryption of the VPN.

3.3.3 The authenticator response

The authenticator response is designed to prove that the server being contacted is indeed trusted. This is done by proving it knows the NT session key, which is in turn based on the password. In much the same way that the server checks the client's identiy, the client verifies that the server knows the NT session key. This is done by applying the SHA1[16] digest function over the session key, and some variable data from the authentication process.

The first step mixes the NTLM response with the secret NT session key, by use of SHA1 and a magic constant:

```
    auth1 = SHA1(concat(NTSessionKey, NTLMResponse,
                        "Magic server to client signing constant"))
```

Magic constants in MSCHAP are not null terminated.

This value (20 bytes in length) is then combined the the challenge, generated above, and another magic constant:

```
    auth2 = SHA1(concat(auth1, challenge,
                        "Pad to make it do more than one iteration"))
```

It is this 20 byte value that is returned to the PPP client, as proof that the server was trusted by its domain controller. (It is returned as an ASCII string, by expanding each byte into two hexadecimal digits).

3.3.4 The MPPE encryption keys

In a similar way to how the MSCHAPv2 authenticator response is generated, encryption keys for the VPN connection must be created, in a way that both sides know, but does not expose the long-term session key. This is done by yet more application of SHA1.

Firstly, a common derivative of the master NT session key is created:

```
    MasterKey = head(SHA1(concat(NTSessionKey, NTLMResponse,
                             "This is the MPPE Master Key")), 16)
```

This master key varies with each authentication, avoiding problems associated with replay attacks on the encrypted connection. Two static padding buffers are used as extra inputs into the system:

SHApad1	40 NULL bytes
SHApad2	40 0xf2 bytes

The master key is then used to generate two sub-keys: one for client-to-server communication, and one for server-to-client communication:

```
ClientToServerKey = SHA1(concat(MasterKey, SHApad1,
                                “On the client side, this is the send key; ”
                                “on the server side, it is the receive key.”,
                                SHApad2))
ServerToClientKey = SHA1(concat(MasterKey, SHApad1,
                                “On the client side, this is the receive key; ”
                                “on the server side, it is the send key.”,
                                SHApad2))
```

These keys are then used for the RC4 encryption of the MPPE stream.

Chapter 4

Integrating a PoPToP VPN server into a Windows Domain

The ability to provide this service from a Linux VPN server was a key requirement for the project. A number of open source packages were integrated into the eventual solution, and new integration work commissioned.

4.1 Base Technologies

4.1.1 PoPToP

PoPToP provides an implementation of PPTP (RFC 2637) on a Linux platform. Its primary purpose is to allow the use of a standard PPP implementation once the extra IP tunneling headers have been stripped off. As soon as the PPTP session is started (and headers removed), PoPToP simply calls pppd to take over the connection.

4.1.2 PPP (Linux PPP implementation)

The Linux PPP implementation required no alteration for use with PoPToP, as ppp version 2.4.2b2 already contained all the required logic for MSCHAPv2 authentication, and could correctly pass off the session keys for MPPE encryption.

4.1.3 MPPE Linux Kernel Module

The MPPE Linux Kernel module provides Microsoft's Point to Point Encryption system. Linux's PPP implementation is shared between user-space and kernel-space. The session is established in user-space, while the kernel actually runs the session. Compression (where the encryption layer is staged) is handled in the kernel.

PPP 2.4.2b3 includes a patch to Linux 2.4.19 kernel to implement the MPPE 'compression' scheme. When applied, the new module `ppp_mppe.o` becomes available.

4.1.4 Samba

Winbind

Samba's Winbind utility forms the vital link in the authentication chain, effectively bridging the gap between a UNIX-like API and the full gamut of Microsoft RPC 'Netlogon' calls. It handles locating the domain controller, all the communication channels and session credentials. Winbind provides its functionality over a unix domain socket. Winbind clients connect to that socket in order to send structured queries.

ntlm_auth

ntlm_auth is a Winbind client that provides a stable interface between the Winbind domain socket (an internal Samba interface) and external programs. Designed in such a way as to be useful in scripts and as a 'helper' to other programs, ntlm_auth's command line arguments should not change over time. Where additional information needs to be communicated to or from the helper, it is done over standard input/ouput.

In particular, ntlm_auth can take user-name, challenge, and the response, and pass them over to Winbind for processing. (Extensions were made and committed to ntlm_auth for this project)

net

'net' provides an interface to allow (amongst other things) Samba to join the domain.

4.1.5 winbind.so PPP module and PPP patches

The only completely new software that was required to be written was the winbind PPP module. The purpose of this module is to link ntlm_auth (and therefore Winbind) into the challenge-response system used inside pppd. The design is heavily based on the radius.so module, which provides similar services, except against a RADIUS server.

In order to properly integrate this software into pppd, patches needed to be applied to expose certain interfaces, which have now been accepted by the PPP maintainer, into PPP 2.4.3.

4.2 Installation

4.2.1 Samba

Samba 3.0.3 (the minimum required version) may be installed, by means of the preferred packaging system for the target platform, or from source. No special compilation options are required, but a source install will usually place its 'prefix' in /usr/local/samba, whereas a binary install will be per the system's normal preferences.

4.2.2 PPP

PPP 2.4.3 is available from `http://ppp.samba.org`, or soon as RPMs and other packaged software.

4.2.3 PoPToP

PoPToP is available as a Debian package, but otherwise is available for download in source and RPM from `http://www.PoPToP.org`.

4.2.4 MPPE Kernel modules

The ppp version specified above assumes the use of the correct kernel modules. The patch is in the ppp/linux/mppe directory, of the source tarball, and assumes a locally-compiled kernel. It is different to the patch marked 'openssl' that has shipped with debian. If your kernel sources were in /data/linux-2.4.22 then you would run:

```
cd ppp/linux/mppe
sh ./mppeinstall.sh /data/linux-2.4.22
```

Hit <return> when prompted, and it should install, possibly with conflicts. If conflicts are found, then examine the .rej files; most are trivial to rectify.

At this point it is a standard kernel compile. The configuration option for MPPE may be found in:

```
Network Device Support -> PPP MPPE compression (encryption)
```

4.3 Configuration

4.3.1 Samba

Samba has a reputation for being hard to configure. In some ways, this reputation is deserved, however it can also be very simple to configure.

The smb.conf configuration file (depending on install method, in /etc/samba or /usr/local/samba/lib) should contain:

For an NT4 or Samba Domain

```
[global]
   workgroup=MYDOM
   security=domain
```

For an Active Directory Domain

(Try the settings listed below. If you experience troubles, it may be easier to 'give up' and use the NT4 compatibility instead. Also read the Samba HOWTO Collection on suggested settings for other configuration files, such as /etc/krb5.conf.)

```
[global]
   workgroup=MYDOM
   realm=MY.ADS.DOMAIN
   security=ads
```

Naturally, replace MYDOM with the 'short' name of your Windows domain, and 'MY.ADS.DOMAIN' with the long (DNS, Realm) name. If your network has a WINS server at (for example, at IP 10.0.0.1) you would be advised to add:

```
   wins server = 10.0.0.1
```

Joining the Domain

The process of joining the domain is very simple; as root run:

```
net join -U administrator
```

If required, net will prompt you for the administrator's password and add your machine to the domain. It will tell you if the join was successful, or what error occurred.

4.3.2 PoPToP and PPP

We will aim not to change the global PPP configuration on the machine we are using as a VPN server. As such, all configuration is in the /etc/pptpd.conf file for PoPToP itself, and /etc/ppp/options.pptpd, for PPP options when invoked under PoPToP.

/etc/pptpd.conf

This file should contain:

```
option /etc/ppp/options.pptpd
localip 192.168.0.1
remoteip 192.168.0.200-254
```

In this file, localip can be either an address in the 'VPN' subnet, or simply the address of the machine's external Ethernet interface. The remoteip specifies the addresses that will be allocated to the clients. This can be at most a range of 254 addresses. Specifying 192.168.10-50.1 will simply allocate the addresses 192.168.10.1, 192.168.11.1 to the remote clients. If the 'remoteip' addresses are actually addresses on the local LAN, then set the 'proxyarp' option in /etc/ppp/options.pptpd.

/etc/ppp/options.pptpd

This file controls the way PPP behaves, and in particular, the authentication and security applied to the link. Take care when considering what options to use.

```
## CHANGE TO SUIT YOUR SYSTEM
lock
## turn pppd syslog debugging on
debug
# The server will prove itself to us, but not the 'normal' way
# so turn that off.
noauth
## change 'pptpd' to whatever you specify as your server name in chap-secrets
name pptpd
# Don't need this
nobsdcomp
# Bring VPN clients onto the local LAN
#proxyarp
# These options are for use with the BSD-licensed patch (ppp => 2.4.2)
# This is the default implementation
refuse-pap
refuse-chap
refuse-MSCHAP
require-MSCHAP-v2
require-mppe
# These options will tell ppp to pass on these to your clients
# To use ms-dns or ms-dns in options.pptpd it must exist in /etc/resolv.conf
ms-wins <ip-of-your-winsserver>
ms-dns <ip-of-your-dnsserver>
# This tells the clients to use us as their default route
defaultroute
# Tell pppd to use Winbind for authentication
#  (modify to suit your installation):
plugin winbind.so
ntlm_auth-helper /usr/bin/ntlm_auth --helper-protocol=ntlm-server-1
```

4.3.3 MPPE Kernel modules

/etc/modules.conf

As the ppp_mppe modules are not always known about by modprobe, you may need to add:

```
    alias ppp-compress-18 ppp_mppe
```

to your modules.conf file. The module itself should load without errors, but may give:

```
    Warning: loading /lib/modules/2.4.20-8/kernel/drivers/net/ppp_mppe.o will taint the
    See http://www.tux.org/lkml/#export-tainted for information about tainted modules
    Module ppp_mppe loaded, with warnings
```

This is normal.

4.4 Starting the services

nmbd, winbindd, and pptpd need to be started at system boot, in that order. The choice of how to do this is dependent on mode of installation, and distribution.

4.5 Testing (Verifying the installation)

4.5.1 Testing Winbind

After starting all the services, the first part of the system to test is winbindd. As root, run:

```
    wbinfo -p
    wbinfo -t
```

Both these commands should succeed. The first checks that Winbind is running, and processing requests. The second checks that it is correctly joined to the domain. If these two pass, then run:

```
    ntlm_auth --username=USER --domain=MYDOM
```

Naturally, replacing USER and DOMAIN with a valid user in your domain. ntlm_auth will prompt for the user's password, and try to authenticate them. If that works, try:

```
    ntlm_auth --username=USER --domain=MYDOM --diagnostics
```

Against a Windows 2000 domain controller, none of the tests should fail. If some do, your system may still be operable as long as the 'NTLM' test passed.

4.5.2 Testing PPP

PPP may be tested, against itself, on the server itself. This allows for basic configuration mistakes to be easily found, and for the integration with ntlm_auth to be tested without the influence of the Windows client. This involves the creation of a number of files:

/etc/ppp/peers/self

```
    noauth
    user abartlet
    debug
    lcp-echo-interval 30
    lcp-echo-failure 3
    pty '/usr/sbin/pppd call self-server'
```

/etc/ppp/peers/self-server

```
192.168.1.200:192.168.1.201
auth
debug
name piglett
notty
require-MSCHAP-v2
```

/etc/ppp/peers/winbind

```
noauth
user abartlet
debug
lcp-echo-interval 30
lcp-echo-failure 3
pty '/usr/sbin/pppd call winbind-server'
```

/etc/ppp/peers/winbind-server

```
192.168.1.200:192.168.1.201
auth
debug
name piglett
notty
require-MSCHAP-v2
plugin winbind.so
ntlm_auth-helper /usr/bin/ntlm_auth --helper-protocol=ntlm-server-1
```

/etc/ppp/chap-secrets

This file needs to contain the plaintext password used for the authentication:

```
# Secrets for authentication using CHAP
# client        server  secret                  IP addresses
abartlet        piglett samba2 *
```

When running the *winbind* tests, the username/password here (and the one in /etc/ppp/peers/winbind) must be a valid username on your Domain.

If you intend to run in 'require-MSCHAP' mode, be sure to test with it. mode.

4.5.3 Testing the MPPE encryption module

If the above tests are successful, test adding

```
require-mppe
```

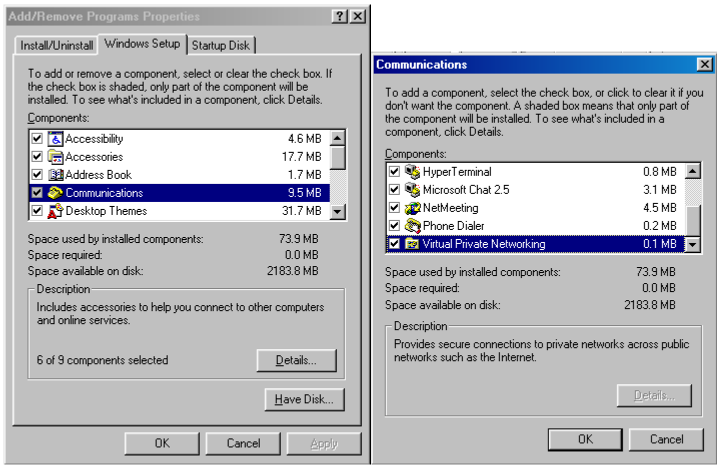
to the peers files, which will then test the MPPE cryptographic module.

Note, as almost all these tests are 'against itself,' many errors (particularly in the cryptographic code) will not show up, as they are perfectly compensated for.

4.6 Client Configuration

4.6.1 Microsoft Windows 98SE

When configuring Microsoft Windows 98, the first step is to ensure that the VPN client software has been installed (figure 4.1). It is generally a good idea to run Windows Update on the computer to ensure all relevant security patches have been applied, however, this step may not be required in all circumstances.


Add/Remove Programs Properties

Install/Uninstall | Windows Setup | Startup Disk

To add or remove a component, select or clear the check box. If the check box is shaded, only part of the component will be installed. To see what's included in a component, click Details.

Components:

☑ Accessibility	4.6 MB
☑ Accessories	17.7 MB
☑ Address Book	1.7 MB
☑ Communications	9.5 MB
☑ Desktop Themes	31.7 MB

Space used by installed components: 73.9 MB
Space required: 0.0 MB
Space available on disk: 2183.8 MB

Description
Includes accessories to help you connect to other computers and online services.

6 of 9 components selected

Communications

To add a component, select the check box, or click to clear it if you don't want the component. A shaded box means that only part of the component will be installed. To see what's included in a component, click Details.

Components:

☑ HyperTerminal	0.8 MB
☑ Microsoft Chat 2.5	3.1 MB
☑ NetMeeting	4.5 MB
☑ Phone Dialer	0.2 MB
☑ Virtual Private Networking	0.1 MB

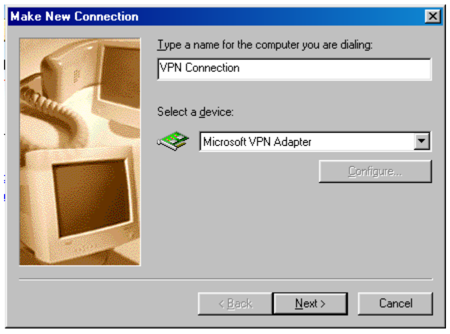
Space used by installed components: 73.9 MB
Space required: 0.0 MB
Space available on disk: 2183.8 MB

Description
Provides secure connections to private networks across public networks such as the Internet.

Figure 4.1: Check that the Microsoft VPN client is installed

Make the connection

New connectionss are configured in the 'Dial Up Networking' window (found under 'My Computer'). Choose 'Make New Connection,' and figure 4.2 should appear. Name the connection, and select 'Microsoft VPN Adaptor' as the device.


Make New Connection

Type a name for the computer you are dialing:
VPN Connection

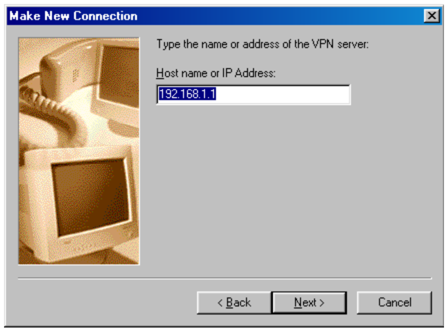
Select a device:
Microsoft VPN Adapter

Figure 4.2: Win98 connection device selection

Then, in the next screen (figure 4.3), enter the hostname or IP address of the VPN gateway. On the wireless network, this should be the IP address of the wireless gateway.

Make the connection

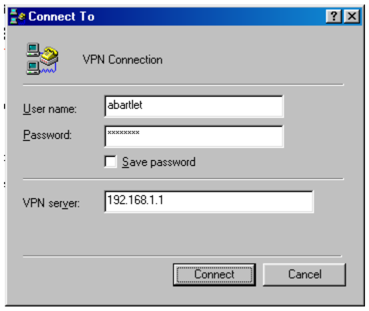
Finally, when the wizard is finished, the connection can be selected from 'Dial Up Networking.' Double-clicking on the icon starts the connection, where it asks for the username and password (figure 4.4).


Make New Connection

Type the name or address of the VPN server:

Host name or IP Address: 192.168.1.1

Figure 4.3: Win98 VPN gateway specification


Connect To

VPN Connection

User name: abartlet

Password: xxxxxxxx

Save password

VPN server: 192.168.1.1

Figure 4.4: Win98 VPN connection

4.6.2 Microsoft Windows XP

Windows XP is very easy to configure as a client, but the wizard is about 8 steps.

Configure the new connection

New connections are set-up by clicking on the link marked 'Create a new connection,' as shown in figure 4.5, on the left-hand side of the 'Network Connections' window. (This can be found under 'Start -> Settings -> Network Connections'). This brings up the 'New Connection Wizard,' shown in figure 4.6.

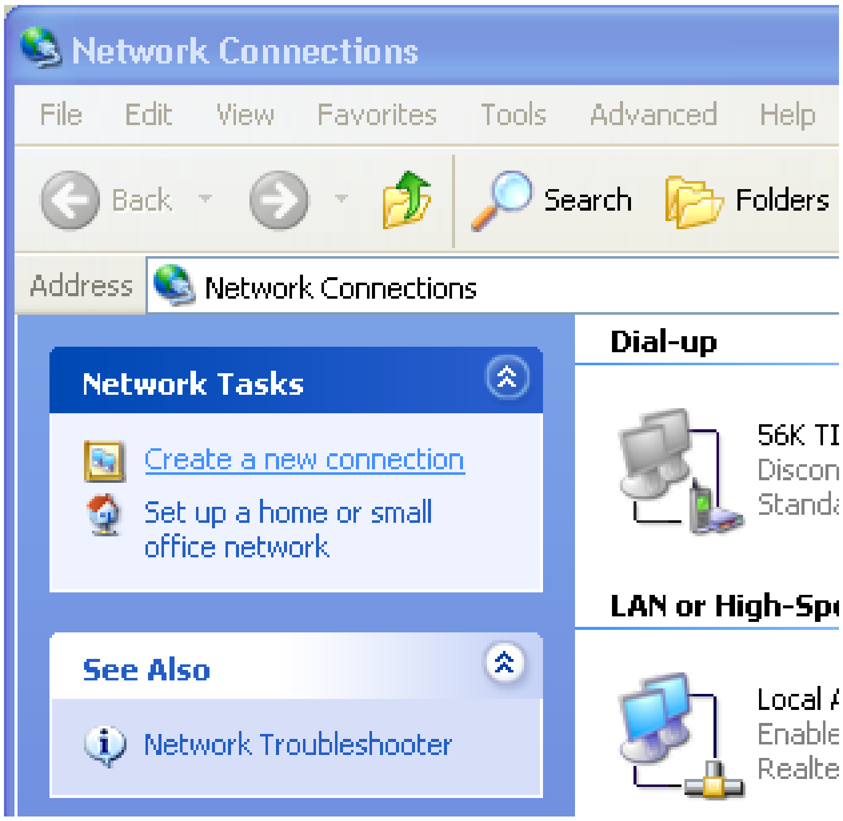
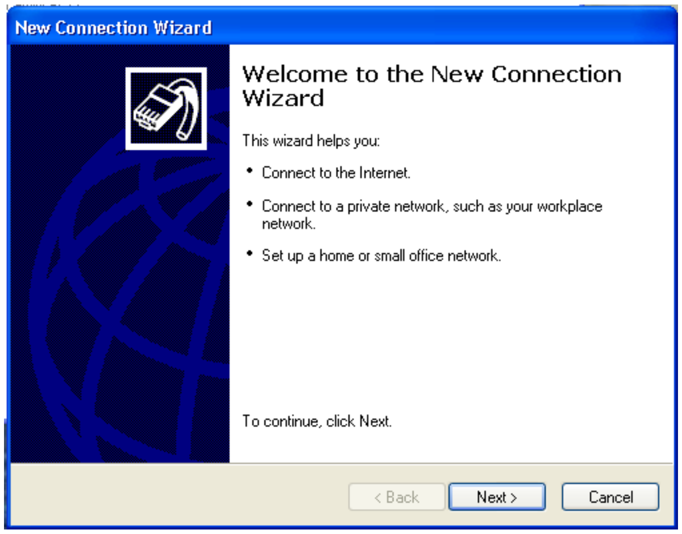


Figure 4.5: Create a new connection



Welcome to the New Connection Wizard

This wizard helps you:

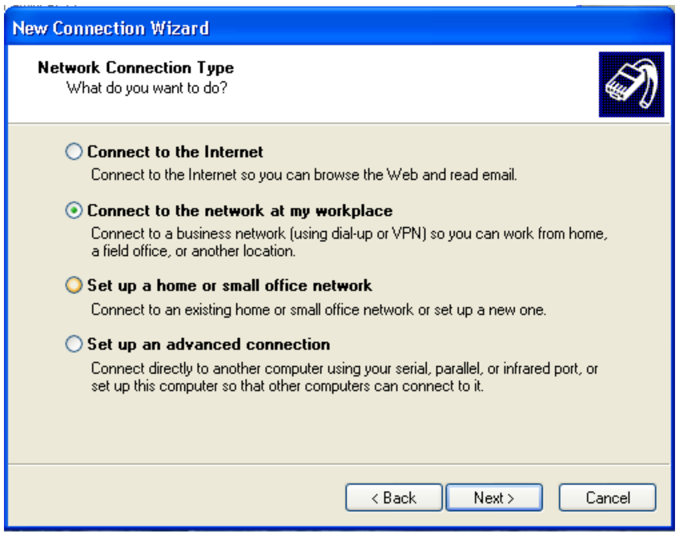
- Connect to the Internet.
- Connect to a private network, such as your workplace network.
- Set up a home or small office network.

To continue, click Next.

Figure 4.6: The 'New Connection Wizard'

Telling Windows to create a VPN connection

The next window allows the selection of a connection type. Select 'connect to a network at my workplace' (figure 4.7).



Network Connection Type
What do you want to do?

( ) **Connect to the Internet**
Connect to the Internet so you can browse the Web and read email.

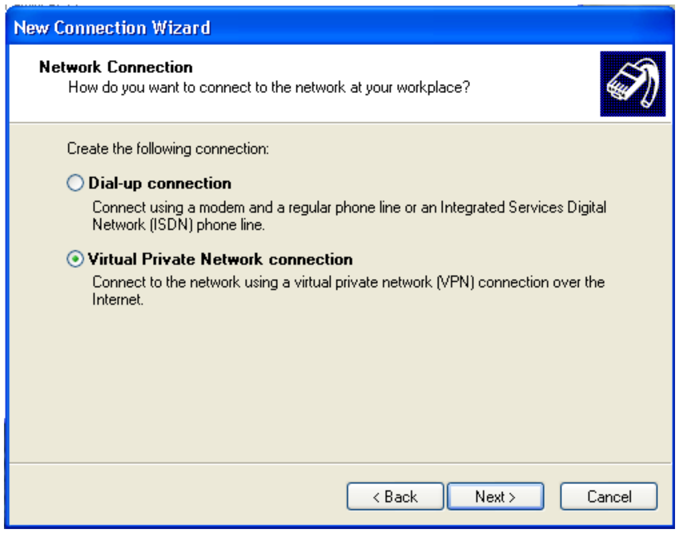
(●) **Connect to the network at my workplace**
Connect to a business network (using dial-up or VPN) so you can work from home, a field office, or another location.

( ) **Set up a home or small office network**
Connect to an existing home or small office network or set up a new one.

( ) **Set up an advanced connection**
Connect directly to another computer using your serial, parallel, or infrared port, or set up this computer so that other computers can connect to it.

Figure 4.7: Select network connection type

Now select 'Virtual Private Network connection,' as show in figure 4.8, and select 'do not dial the initial connection' on the next screen (figure 4.9).



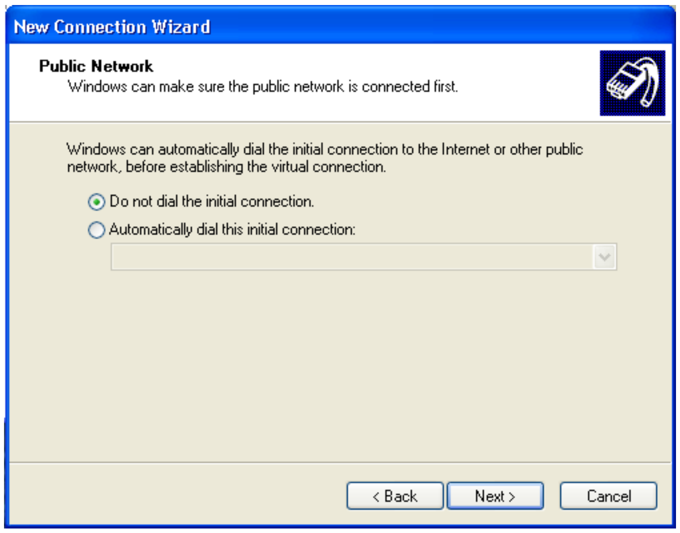
Network Connection
How do you want to connect to the network at your workplace?

Create the following connection:

( ) **Dial-up connection**
Connect using a modem and a regular phone line or an Integrated Services Digital Network (ISDN) phone line.

(●) **Virtual Private Network connection**
Connect to the network using a virtual private network (VPN) connection over the Internet.

Figure 4.8: Select connection method


New Connection Wizard

Public Network

Windows can make sure the public network is connected first.

Windows can automatically dial the initial connection to the Internet or other public network, before establishing the virtual connection.

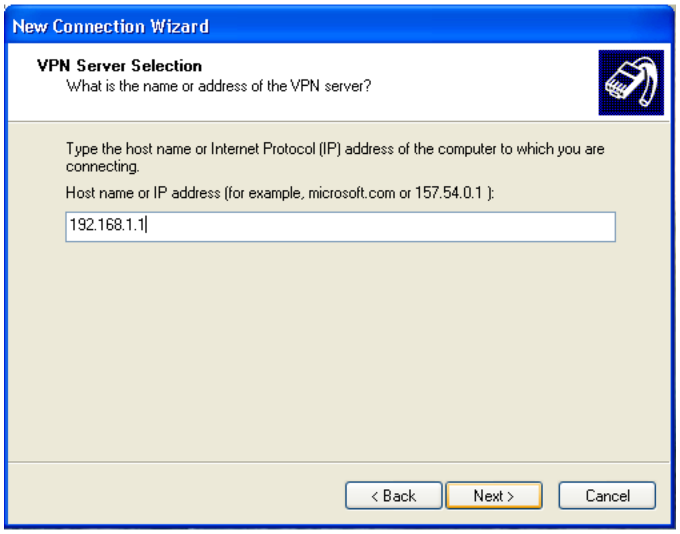
(●) Do not dial the initial connection.

( ) Automatically dial this initial connection:

Figure 4.9: Connect to a public network first?

Locate VPN server

Finally, the VPN wizard needs to know where the VPN server is. The next screen, shown in figure 4.10, takes either a DNS name, or an IP address. On the wireless network, this should be the IP address of the wireless gateway. When this is completed, so is the wizard (figure 4.11)


New Connection Wizard

VPN Server Selection

What is the name or address of the VPN server?

Type the host name or Internet Protocol (IP) address of the computer to which you are connecting.

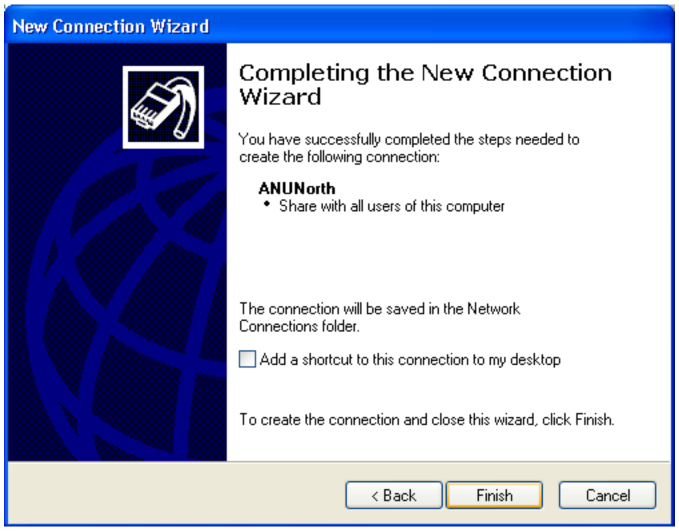
Host name or IP address (for example, microsoft.com or 157.54.0.1):

192.168.1.1

Figure 4.10: VPN server address

Connecting

The next screen, after the completion of the wizard, is the login screen (figure 4.12). Simply enter a valid username/password for the VPN server, and click 'Connect.'



New Connection Wizard

Completing the New Connection Wizard

You have successfully completed the steps needed to create the following connection:

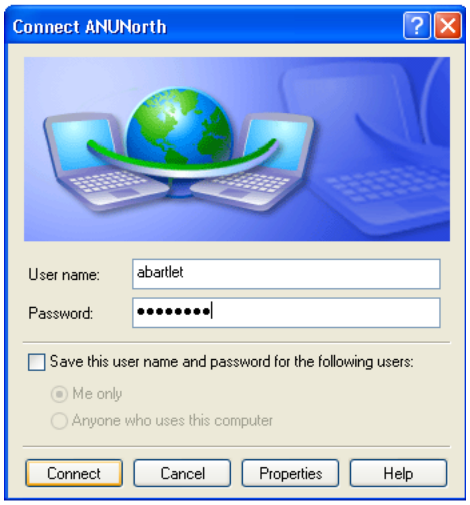
ANUNorth
- Share with all users of this computer

The connection will be saved in the Network Connections folder.

☐ Add a shortcut to this connection to my desktop

To create the connection and close this wizard, click Finish.

Figure 4.11: Finished!



Connect ANUNorth

User name: abartlet

Password: ••••••••

☐ Save this user name and password for the following users:

● Me only

○ Anyone who uses this computer

Figure 4.12: Connection Dialog

Diagnosis

Both the client and the server can provide a reasonable amount of information on an invalid connection, but the most common mistake is the use of invalid configuration directives. Check these first, then check the system with only 'local' authentication (no winbind/ntlm_auth).

4.6.3 Linux

Linux is by far the most difficult of the 3 clients to configure. Basically, all the steps shown above for the server's pppd installation must be taken. There is no need for PoPToP, but another package - pptp-client[22] must be installed, as must the correct ppp_mppe kernel modules. A HOWTO is available on the pptp-client website.

Chapter 5

Design and implementation

5.1 Overall design considerations

In designing a solution for this problem, a strong emphasis was placed on code reuse: This project is a systems intergration project, not a software development project. In many ways, this hampered what might have been more ideal design choices, but allowed for much greater reuse of existing infrastructure, code, and protocol understanding. It is this chapter that describes the new code produced, and new functionality obtained.

5.1.1 Seperation of authentication components

It was decided to follow Microsoft's lead, and separate the authentication database from the VPN server. This has the particular advantage that a successful attack on the VPN server will not yield user passwords, nor do we need to maintain a separate user-password database. (Indeed, it was to allow for this that many of MSCHAP and MSCHAPv2's fundemental design compromises were made).

Choosing this authentication chain design has a particular benefit: It has been used with Squid to authenticate clients using Microsoft's NTLMSSP authentication. As such, the interfaces required were already present, simply needing adaptation to a new and slightly different problem.

5.1.2 Alternative Designs

During the implementation phase of this project, it became clear that there was an alternative design. Instead of implementing a winbind plugin for PPPd, the intergration could have occured on a RADIUS server. (A RADIUS server is often used to check passwords on behalf of a dial-in

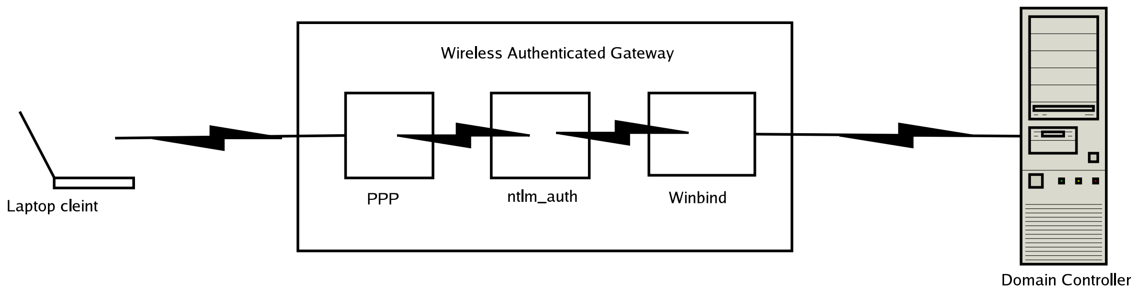


Figure 5.1: The building blocks in the authentication chain

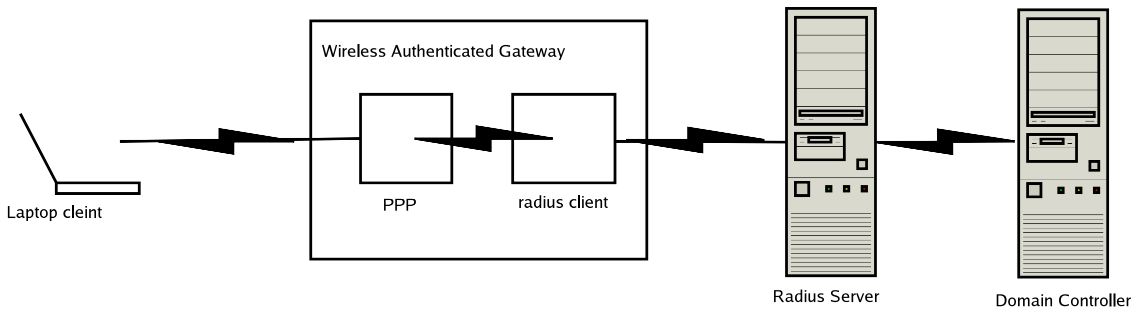


Figure 5.2: A Radius-based VPN gateway

Similarly, RADIUS[1] could be used link the gateway to the network. The RADIUS server could well be the same machine as the DC.

modem server, or similar). A RADIUS server is available for use in Active Directory, and it now possible to link the Open Source FreeRADIUS server into NT domains, using `ntlm_auth`.

5.2 The role of Samba

Samba's role in this project is to join the domain, and provide the services that other tools can use. In particular, Samba's role is to ensure that the other tools only deal with authentication, not with contacting the Domain Controller.

5.2.1 Joining and using domain

Joining an NT4 domain is a complex, multi-step process. Samba's 'net join' command will establish it, and a number of Samba components can use it. While whole books can and have been written on the processes involved, it can be described as:

Joining

This process sets up a shared secret between the Domain Controller and the Samba 'Member Server.' The server is a 'member' in that it is registered with the DC as having a particular name, and a particular shared secret (the 'machine account password').

Contacting the Domain Controller

Once joined, the Domain Member needs to locate the domain controller (using the native windows protocols) and establish its identity with the DC.

Authenticating users

The member server uses the connection, and the result of its authentication to the DC to securely pass the user-name and password of users attempting logins. The 'secure channel' is typically encrypted, and ensures that the information about the user (including the password-derived 'session key') is not compromised in the reply.

5.2.2 Samba components

Winbind

Winbind's fundamental purpose is to provide Samba's 'NT to Unix' infrastructure, particularly in the area of name services and authentication. It also manages the connections to the domain controller.

ntlm_auth

This is a Winbind client program, provided as part of Samba 3.0. This program works with Winbind to authenticate users. Winbind provides an interface to authenticate an arbitrary username/password, and ntlm_auth translates that interface into the one that programs like Squid, Apache or pppd can use. Because Winbind handles all the connections to the domain controller, ntlm_auth is a small, easily understood and easily extended client.

pam_winbind

Another Winbind client, this is a PAM module that allows programs taking plain-text passwords to pass them on to the Domain Controller for authentication. While this is the traditional winbind authentication client, it is only useful for plaintext passwords, and so plays no part in this system.

smbd

Similarly, Samba's main File-server component plays no part in the authentication of Unix users, unless Samba is acting as a PDC as well.

5.3 Building blocks

The following existing programs were modified for this project:

5.3.1 PoPToP

The PoPToP source was unmodified and installed in the usual manner. The integration with ppp was as per its documentation.

5.3.2 PPP

PPP was modified to allow the use of the winbind plug-in. This plug-in required access to some further internal functions, and, as such, a patch was developed.

5.3.3 Samba 3.0

Components of Samba 3.0 were used by (and modified for) this project. All these modifications have been included in release of Samba 3.0.3.

Winbind

Winbind was unmodified, and installed normally.

`ntlm_auth`

`ntlm_auth` was modified to allow user-names and challenge-response passwords to be passed as command-line options. This made the development of the winbind.so plug-in much easier.

5.3.4 The winbind.so pppd plug-in

The winbind.so plug-in was written from the skeleton of the radius plug-in. It interfaces with ntlm_auth via the command line.

5.3.5 The domain controller

The domain controller is unmodified for this project, but must meet certain requirements. In particular, it must support the NT user-session-key, which is a value returned from a successful password authentication. The MSCHAP and MSCHAPv2 systems use this user-session-key. Samba 2.2 did not support this, but it is standard in Samba 3.0, and in all recent Microsoft implementations (including NT 4.0).

5.4 Designing the extensions

5.4.1 ntlm_auth

`ntlm_auth` is a Samba utility intended to provide a consistent interface to Winbind. Winbind operates over a domain socket, but the protocol in use on that socket varies between versions of Samba. In the past, external projects have learnt how to use this interface, and are now faced with nasty compatibility issues caused by Samba 3.0's changes. This particularly affected the Squid project, which was using winbind to validate NTLMSSP authentication for Microsoft's Internet Explorer browser. To avoid a repeat, ntlm_auth was created, to provide a stable interface that any number of external projects could depend on.

`ntlm_auth` operates as a normal Unix command-line program. It uses command line options to switch between various operating modes, and passes data in and out via the standard input and standard output. It is designed in such a way that future extensions should not impact on existing users.

The changes

For this project, ntlm_auth needed to be extended to take user-names and 'challenge/response' passwords outside an NTLMSSP wrapping. Similarly, since MSCHAP and MSCHAPv2 requires use of the NT session key; ntlm_auth needed to return them in a useful way.

To achieve this, new command line options were added:

`--username` The remote user's name

`--domain` The remote user's domain

`--nt-response` Specifies the NTLM response

`--challenge` Specify the challenge for input into the LM response algorithm

`--request-nt-key` Return the user's NT session key on the standard output

When operated in this mode, `ntlm_auth` indicates success or failure simply by its return code.

For the second generation patch, (included with this report), another option was added:

`--helper-protocol=ntlm-server-1` Provide a Standard Input / Standard Output based system for the provison of the username challenge, and response variables.

This mode, included in Samba 3.0.3, avoids the problems associated with command-line arguments, particularly when the values presented may be malicious, or sensitive. While the existing pppd patch design did not suffer explict problems in this area, it was clear that other implementations might not be so careful.

5.4.2 Designing the winbind.so plugin

The `pppd` plugin interface

`pppd` provides an plugin interface by which it's functionality may be modified - a plugin may alter behaviours such as IP address selection, as well as authentication. If a plugin is configured, `pppd` will load and call it, with the system challenge and response values as arguments. The actual interface is a loadable .so (unix shared object), which fills in certain function pointers upon its initialisation. Quite simply, if the function-pointer is non-null, the function is called.

An example of a working challenge-response plug-in is already provided, in the form of the radius plug-in. Indeed, many of the extensions to the plug-in interface needed by the `winbind.so` plugin were already provided. (Other extensions were required and added, and these are included in a patch to `pppd`.)

The changes

The winbind.so plug-in needs to interface between `pppd` and `ntlm_auth`, in the same way that the existing radius plugin links `pppd` to the radius client libraries. As the published interface is the `ntlm_auth` executable, a `fork()/exec()` design was implemented:

The main `pppd` process is forked into two processes, and the child executes the `ntlm_auth` helper, with the appropriate command-line options. The parent waits for it to return, and reads its standard output for the user's NT session key. If the `ntlm_auth` process returns success, the user is authenticated, and the user session key is transformed into the return authenticator and the MPPE encryption keys. By reading these details over the standard output, the communication between the parent and the child is secure.

5.4.3 The patches

Attached to this report are two 'patch' files, processed so as to be more understandable, that show not only the final product, but also the development process used to obtain it. The patch '`pppd-ntlm-auth.patch`' comprises the complete changes required to make the current `pppd` code use `ntlm_auth` for authentication. And electronic copy of the patch is applied, per the instructions in this report.

More interesting however, is the patch between '`radius.c`' and '`winbind.c`'. This patch, '`radius-winbind.patch`' shows how the development of this project avoided re-inventing the wheel. The `ntlm_auth` modal of authentication is very much like that used by radius - in both cases the remote server returns 'sucess' or 'failure', and in both cases the remote server returns the session keys, in some form or other. As such, the new pppd plugin was written not from scratch, but by simply exchanging the glue code. The function prototypes were maintained, as they belong to the interface, but the internals simply made different calls.

This shows one of the major benifits of the Open Source approach to software design - that these incremental changes can be built on the infrastructure provided by others.

Aside from some minor changes inside `pppd` (due to the need to expose some extra interfaces to the winbind.c plugin), it will be noted that all the new logic is self-contained, inside this plugin. This has meant that the plugin patch was successfully maintained for almost 12 months, without major impact from internal pppd restructuring.

Appendix A

Windows Authentication

For many years now, the stalwart of Windows/Unix compatibility has been Samba - the file, print and domain controlling server distributed by the Samba Team.

Traditionally, Samba has only been able to help with some of these problems: the problems confronted in running a file server. This paper shows how Samba can be used beyond the file-server, integrating with other software to provide *integrated windows authentication*.

A.1 What is Windows Authentication?

Microsoft Windows authentication has a long history. Originally based on the file and print authentication technology embodied in the SMB protocol[1] and Microsoft's LAN Manager network servers, it has grown at the heart of Windows authentication today. Indeed, windows authentication has now been integrated with the Kerberos network authentication protocol. Kerberos plays no part in the technology discussed in this document, which all relates to NT4 compatible authentication.

A.1.1 Properties of Windows Authentication

One of the strongest properties of Windows authentication is that is must be able to be carried out over an untrusted network, without revealing the password, and that servers participating in the network must have some way to verify the password, without access to the password itself.

This usually means that it becomes the job of the Domain Controller (DC) to verify passwords passed on by others.

Another property of windows authentication is that the way the DC is contacted has not changed over the entire life of Windows NT 4.0, and that Windows 2000 and Windows 2003 have remained compatible with that. This means that inventions of new authentication protocols are often reworked older versions, so as to require as few software upgrades as possible.

A.1.2 Domain and Transparent Authentication

The other very strong property of 'Windows Authentication' is that the concept of a 'network login' - the Single Sign On. No matter how it is actually implemented, Windows Authentication always appears as if the user 'logs in' to the network, and is automatically authenticated to any of the co-operating servers. For NTLM, this is handled by caching the user's password, but as the user doesn't need to know this, they don't care.

[1]Server Message Block, later known as CIFS, Common Internet File System[1]

A.1.3 Other Windows Authentication Schemes

There are a number of somewhat compatible challenge-response authentication schemes used by Windows servers:

- LM
- NTLM
- NTLMv2
- LMv2

These are all described in detail by 'Implementing CIFS'[1]. Each provides a way for a client to authenticate to a server, in a way that will not reveal the password to the network. (That said, the LM form is very weak cryptographically).

A.1.4 NTLMSSP

NTLMSSP is Microsoft's general user authentication protocol, operating on the basis of message exchange, rather than a specific protocol that applications must implement. This means that the same message exchange may occur over HTTP, SMB, IMAP etc. Documentation about this protocol has been hard to find, but is being collected at 'http://davenport.sf.net/ntlm.html'.

Because of the interface provided to programmers by SSPI (Security Support Provider Interface) on a Microsoft platform, where there is no compelling reason to do otherwise, Microsoft has simply merged NTLMSSP into various protocols, rather than create an authentication scheme more in line with existing practice.

A.1.5 Kerberos

Probably the biggest change in the Windows Authentication landscape since the introduction of the domain modal in NT 3.5 was the introduction of Kerberos in Windows 2000, as part of Active Directory. This has resulted in changes to a large number of network protocols - most of those that supported NTLMSSP, now also support Kerberos.

A.2 Existing Projects

A large number of existing projects have attempted (with varying degrees of success) to integrate authentication with Windows domains, using these technologies. They can be grouped into NTLMSSP, and 'other,' in that most projects (like most of the protocols involved) use NTLMSSP wrappings on their authentication.

A.2.1 Samba's Winbind

Samba has, since version 2.2.3 provided Winbind, to enable user lists and PAM-based authentication to be made to a Windows DC. For many years now, Samba has had the ability to authenticate SMB connections against a windows DC.

A.2.2 NTLMSSP-based projects

Squid

Squid (www.squid-cache.org) has its own NTLMSSP-over-HTTP implementation, which has a number of back-ends with which to authenticate proxy users. The 'helper' design has allowed Samba to provide a compatible helper, using its own NTLMSSP code.

Apache

mod_auth_ntlm (modntlm.sf.net) is a project to enable NTLMSSP-over-HTTP authentication to the Apache 1.3 web server. It includes its own NTLMSSP implementation.

Fetchmail

Fetchmail includes a client-side implementation for NTLMSSP-over-IMAP.

A.2.3 Other Integrated Authentication projects

pam_krb5

As a standard Kerberos server, Microsoft's Active Directory Services (ADS) can be used with pam_krb5.

pam_ldap

Likewise, 'ldap authentication' can also be used against the LDAP component of ADS.

pam_smb

This pam module makes an SMB connection to the target 'authentication server' and attempts a login. This demonstrates that the user-name/password is correct.

pppd + radius plugin

pppd includes MSCHAP (v1 and v2) integration, via the radius server than Microsoft provides for use with ADS.

FreeRADIUS + ntlm_auth

FreeRadius now includes a patch, very similar to the patch presented here for pppd, to perform the same task at the RADIUS server.

Bibliography

[1] Hertel, Chris 2003, Implementing CIFS, Prentice Hall. ISBN 0-13-047116-X. Also online retrieved: 2 November 2003, from `http://www.ubiqx.org/cifs`.

[2] Borisov, N., Goldberg, I. & Wagner, D (n. d.), *Security of the WEP algorithm,* Retrieved: 2 November 2003, from `Bibliographyhttp://www.isaac.cs.berkeley.edu/isaac/wep-faq.html`.

[3] *RFC Editor Webpage* 2004, Retrieved: 29 April 2004 from `http://www.rfc-editor.org`

[4] IETF 2003, *IP Security Protocol (ipsec)* Charter, Retrieved: 2 November 2003, from `http://www.ietf.org/html.charters/ipsec-charter.html`.

[5] Titz, Olaf 2003, *CIPE - Crypto IP Encapsulation*, Retrieved: 2 November 2003, from `http://sites.inka.de/sites/bigred/devel/cipe.html`.

[6] Microsoft Corporation 2001, *Point-Point Tunneling Protocol (PPTP) FAQ,* Retrieved: 2 November 2003, from `http://www.microsoft.com/ntserver/ProductInfo/faqs/PPTPfaq.asp`.

[7] Schneier B, & Mudge 1998, *Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)*, Retrieved: 2 November 2003, from `http://www.schneier.com/paper-pptp.html`.

[8] Schneier B, & Mudge 1999, *Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2)*, Retrieved: 2 November 2003, from `http://www.schneier.com/paper-pptpv2.html`.

[9] Hobbit, 1997, *CIFS: Common Insecurities Fail Scrutiny,* Retrieved: 2 November 2003, from `http://www.signaltonoise.net/library/cifs.htm`.

[10] Zorn, G., & Cobb, S. 1998, *Microsoft PPP CHAP Extensions, RFC 2433*, Retrieved: 2 November 2003, from `http://www.ietf.org/rfc/rfc2433.txt`

[11] Zorn, G. 2000, *Microsoft PPP CHAP Extensions, Version 2, RFC 2759*, Retrieved: 2 November 2003, from `http://www.ietf.org/rfc/rfc2759.txt`

[12] Pall, G., & Zorn, G. 2001 *Microsoft Point-To-Point Encryption (MPPE) Protocol, RFC 3078*, Retrieved: 2 November 2003, from `http://www.ietf.org/rfc/rfc3078.txt`

[13] Microsoft, Corporation 1999 *Windows 98 Dial-Up Networking Security Upgrade Release Notes*, Retrieved: 2 November 2003 from `http://support.microsoft.com/support/kb/articles/Q189/7/71.asp`.

[14] Unicode Consortium, The 1996 *The Unicode Standard, Version 2.0*, Addison-Wesley, 1996. ISBN 0-201-48345-9.

[15] Rivest, R. 1992, *MD4 Message Digest Algorithm, RFC 1320.* Retrieved: 2 November 2003 from `http://www.ietf.org/rfc/rfc1320.txt`.

[16] National Institute of Standards and Technology 1995, *Secure Hash Standard, Federal Information Processing Standards Publication 180-1.*

[17] RSA Laboritories 2003, *What is RC4?* Retrieved: 2 November 2003 from `http://www.rsasecurity.com/rsalabs/faq/3-6-3.html`.

[18] Mantin, Itsik, *RC4 Page,* Retrieved: 2 November 2003 from `http://www.wisdom.weizmann.ac.il/~itsik/RC4/rc4.html`.

[19] Schneier, Bruce 1996, *Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C* John Wiley & Sons, Inc.

[20] *IEEE P802.11, The Working Group for Wireless LANs* 2003, Retrieved: 2 November 2003 from `http://grouper.ieee.org/groups/802/11/`.

[21] *Wi-Fi Alliance Index* 2003, Retrieved: 2 November 2003 from `http://www.weca.net/OpenSection/index.asp`.

[22] PPTP Client (n. d.), Retrieved: 2 November 2003 from `http://pptpclient.sourceforge.net/`.