AlliedWare

How To | Create an X.509 Certificates VPN Between an Allied Telesis Router and a Windows XP Client

Introduction

In commonly deployed IPsec VPN solutions, ISAKMP (Internet Security and Key Management Protocol) is used to dynamically establish the Security Associations and provide key management for VPN encryption. Public key techniques or, alternatively, a pre-shared key, are used to mutually authenticate the communicating parties.

This solution uses X.509 certificates to provide a public key technique to seed the ISAKMP negotiation between the VPN access concentrator (AT router) and VPN remote host (XP Computer), thus providing a more secure authentication method than the alternate practice of using a pre-shared key. For security, the certificates must be signed by a trusted third party - the Certificate Authority. In addition, at the PPP level of this VPN solution, users are authenticated using CHAP authentication to a user database.

List of terms:

RSA Key Pair

A Key Pair are the encryption keys in an Asymmetric encryption algorithm. The pair contains a public and a private key. The special properties of the keys is that data which is encrypted with the public key can be decrypted with the private key. A device will distribute its public key to other devices, but keep its private key secret. Other devices will use the public key to encrypt data being sent to the holder of the private key. RSA is a popular asymmetric encryption algorithm.

Digital Certificate

A document that states who owns a given encryption key. The document contains the digital signature of a mutually trusted certificate authority.

Digital Signature

A number that can be appended to any electronic document. The number is an algorithmic hash of the document contents, that is then encrypted with a unique encryption key.

What information will you find in this document?

This How To Note begins with the following information:

- "Related How To notes" on page 2
- "Which products and software version does it apply to?" on page 2

Then it describes the configuration, in the following sections:

- "Solution setup" on page 5
- "Solution task details" on page 8
- "Caveat statement" on page 51
- "Appendix" on page 52

Related How To notes

VPN how to notes

Allied Telesis offers How To Notes with a wide range of VPN solutions, from quick and simple solutions for connecting home and remote offices, to advanced multi-feature setups. Notes also describe how to create a VPN between an Allied Telesis router and equipment from a number of other vendors.

For a complete list of VPN How To Notes, see the *Overview of VPN Solutions in How To Notes* in the How To Library at www.alliedtelesis.com/resources/literature/howto.aspx.

Which products and software version does it apply to?

This Note applies to the following Allied Telesis routers and managed Layer 3 switches:

- AR415, AR440, AR441, AR442, and AR700 series routers.
- Rapier series switches
- Software versions: 291.19+

Solution overview

Hardware and software used in this solution

This solution was tested on an AR442 running software maintenance release 291-19. Also, Linux version 2.6.8.1-12mdk and OpenSSL version 0.9.7d were used.

Here are the commands we used to check what version of AR442, Linux, and OpenSSL, were running:

- `SecOff Certificate VPN demo> show install`

```
Install     Release              Patch                GUI
-------------------------------------------------------------------
Temporary   -                    -                    -
Preferred   flash:54291-19.rez   -                    -
```

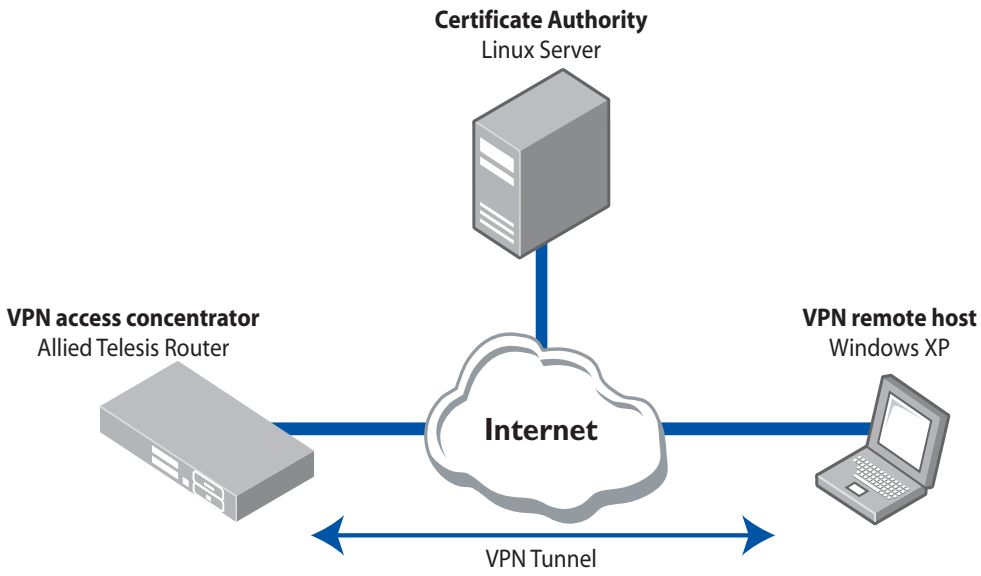
- `[root@multibox URLcerts]# uname -r 2.6.8.1-12mdk`
- `[root@multibox ca]# openssl version OpenSSL 0.9.7d 17 Mar 2004`

The certificates required in this solution are for the following roles:

1. Certificate Authority (CA) - this is the trusted certificate
2. VPN access concentrator - Allied Telesis router (validated by CA)
3. VPN remote host - Windows XP client (validated by CA)

The **Certificate Authority** role can be provided by a Linux Server, a Windows Server, or a third party Certificate Authority paid service.

This solution uses a **Linux** Server as the Certificate Authority.



Security features

The security features in this solution include:

Encryption

Encryption provides data confidentiality, preventing interpretation of captured packets without the encryption key.

User and data origin authentication

- Public key certificates along with IPsec ESP provide a convenient, reliable method for verifying the identity of a sender, and provide per-packet data origin authentication.
- Centralized authentication and accounting can be provided with RADIUS.

  L2TP/IPSec connections provide stronger authentication by requiring both computer-level authentication through certificates and user-level authentication through a PPP authentication protocol.
- Source Address Management.

Data integrity

Data integrity (proof that the data was not modified in transit).

Attack protection

- PPP authentication exchange is encrypted making off-line dictionary attacks much more difficult.
- Replay protection (prevention from resending a stream of captured packets).

Solution setup

The following paragraphs describe the process of setting up a certificate VPN solution. It is not possible to set up one device in it's entirety before moving to the next device. Some steps have pre-requisite steps on another device. Given that the sequential steps involve moving from one device to an other, each step has been **colour coded** to help you identify which device the step applies to.

Color coding

The color codes for each device are as follows:

Certificate Authority (CA)	VPN access concentrator - Router	Windows XP - VPN remote host

Solution tasks - sequential steps

Perform the tasks as listed in the tables below. Each task is described in detail at the end of this section in "Solution task details" on page 8.

On the CA	
Task **A**	Locate OpenSSL tool and set up directories.
Task **B**	Generating the CA's Public /Private Key Pair and its own public X.509 certificate.
Task **C**	Make a copy of certificate file with the file extension expected by the router. Transfer certificate files to a directory ready for transfer to router and XP host.

On the Router	
Task **D**	Creating a Security Officer user and enabling System Security mode.
Task **E**	Adding the CA's public certificate into the PKI Certificate database and setting as trusted.
Task **F**	Generating a RSA Public /Private Key Pair.
Task **G**	Generating a Certificate Signing Request for the router's own certificate.

On the CA	
Task **H**	Receiving a Certificate Signing Request from the router, which it will sign to produce the router's public certificate.

On the Router	
Task **I**	Adding the router's own certificate into the PKI Certificate database, and checking it is validated and trusted.
Task **J**	Configuring the network and a VPN Server facility.
Task **K**	Configure a firewall.

On the VPN remote host	
Task **L**	Set-up an MMC Certificates Console to manage your certificates.
Task **M**	Download the CA's Certificate to the Windows XP - VPN remote host.
Task **N**	Import the CA Certificate to the Trusted Root Certification Authority Store.

On the CA	
Task **O**	Generating a Certificate Signing Request on behalf of the Windows XP - VPN remote host.
Task **P**	Produce the Windows XP Public Certificate which the Windows XP - VPN remote host can import to its Certificate Registry.
Task **Q**	Converting the Windows XP certificate to a form that includes a Private Key.

On the VPN remote host	
Task **R**	Download and import the Windows XP-VPN remote host Certificate to the Certificate Registry.
Task **S**	Setting Up the Windows XP Client VPN Network Connection.

On the Router	
Task **T**	Verification of VPN Connection from Router viewpoint.

Summary of trust relationship that the certificates create

Just for clarity, here is a brief overview of what is being done by the above tasks to get the right certificates in the right places, and how that enables the VPN access concentrator and the remote VPN client to trust each other's encryption keys.

1. Create a root CA certificate on the Certificate Authority server. (Task B)
2. Load the root CA certificate into the **Trusted Root CA** certificate lists on both the VPN access concentrator and the remote VPN client, so they both agree that they trust this CA. (Task B and M)
3. On the VPN access concentrator, create an RSA keypair, and then create an unsigned certificate to verify the VPN access concentrator's ownership of that RSA keypair. (Task F and G)
4. Upload the unsigned certificate to the CA server. The CA server signs the certificate, and the new, signed, version of the certificate is loaded back onto the VPN access concentrator. (Task H and I)

5. The VPN access concentrator now has a certificate, verifying its ownership of its RSA keypair, signed by a certificate authority that the remote VPN client also trusts.
6. On the CA server, create an RSA keypair, and a signed certificate for that keypair, on behalf of the remote VPN client. (Task O and P)
7. Combine the RSA keypair and the signed certificate into a single file, and load them onto the remote VPN client. (Task Q and R)
8. The remote VPN client now has its own RSA keypair, and a certificate verifying its ownership of that RSA keypair. The certificate is signed by a CA that the VPN access concentrator trusts.
9. Then, when the VPN access concentrator and remote VPN client want to exchange encrypted data, they will send their public encryption keys to each other. When each device sends data to the other, it will be expected to encrypt the data using the public key it received from that other device. The VPN access concentrator and remote VPN client each trust that the other device is the true owner of the public key that it sent, as they also send each other certificates that verify that they are the true owners of those RSA encryption keys. The VPN access concentrator and remote VPN client trust each others' certificates because the certificates are both signed by the CA server, and they have both loaded the CA server's certificate into their list of trusted CAs.

Solution task details

On the CA	
Task A	**Locate OpenSSL tool and set up directories** At the Linux BASH prompt ensure you have access to the openssl tool: “whereis openssl" or "locate openssl", after "updatedb" This solution was documented using version 0.9.7a: [root@localhost certificates]# openssl version OpenSSL 0.9.7a Feb. 19 2003 If you cannot locate the openssl tool it may need to be installed on your system. Once installed "man openssl" provides the manual describing usage of the openssl tool. It is recommended that you create a directory for handling certificates, and another for handling the CA's certificates: "mkdir certificates", "cd certificates", "mkdir ca"

Task B **Create the CA's Public /Private Key Pair and its Public X.509 certificate**

Once created, this certificate can be freely distributed to the VPN peers and any other security device (member entity) who trusts this CA and whose certificates will be signed by this CA. The CA's certificate contains the CA's public key.

Create the CA certificate:

Use the **OpenSSL tool**, (as described in Task A, change to the "certificates" directory with "ca" as the child directory).

```
[root@multibox URLcerts]# openssl req -x509 -new -out ca/cacert.crt
-keyout ca/cakey.key -days 9999
```

This will start a sequence of prompts, an example of which is given here:

```
[root@multibox URLcerts]# openssl req -x509 -new -out ca/cacert.crt
-keyout ca/cakey.key -days 9999
Generating a 1024 bit RSA private key
.............++++++
.....++++++
writing new private key to 'ca/cakey.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Domain Component 1 (e.g. nz) [nz]:
Domain Component 2 (e.g. co) [co]:
Domain Component 3 (e.g. alliedtelesis) [alliedtelesis]:
Country Name (2 letter code) [NZ]:
State or Province Name (full name) [Canterbury]:
Locality Name (eg, city) [Christchurch]:
Organization Name (eg, company) [YourCompany]:
Organizational Unit Name (eg, section) [OUN]:CSG
Common Name (eg, YOUR name) [YourName]:CA-Cert
Email Address []:.
```

You will note the OpenSSL tool can be used to prompt for other Distinguished Name fields such as the Domain Component fields seen above. Your installation of OpenSSL might not be customised to prompt for these fields. You can make OpenSSL prompt these fields by adjusting the configuration file. Please see the appendix at the end of this document for an example of how to do that.

Confirm that the certificate file has been created:

[root@multibox URLcerts]# ls ca -lct

total 8

-rw-r--r-- 1 root root 1444 May 21 14:21 cacert.crt

-rw-r--r-- 1 root root 963 May 21 14:21 cakey.key

Use OpenSSL to view the fingerprint for confirmation purposes:

```
[root@multibox URLcerts]# openssl x509 -in ca/cacert.crt -SHA1 -
fingerprint|more
SHA1
Fingerprint=75:DE:E3:C5:66:93:1D:35:6A:B1:C5:B2:A2:53:64:E3:2B:D9:9
7:73
```

Task C **Make a copy of certificate file with the file extension expected by the router. Transfer certificate files to a directory ready for transfer to router and XP host.**

On the router, the certificate file must have a .cer extension.

- Make a copy of the file with the .cer extension.
- Move or copy the file to the appropriate TFTP, FTP, or ZMODEM directory, ready for download to the router and to the Windows XP client:

```
[root@multibox URLcerts]# cp ca/cacert.crt /root/cacert.cer
```

The relevant cacert file can then be transferred to the Windows XP Computer, using a file transfer protocol such as TFTP, FTP, or ZMODEM.

Examples of loading the certificate onto the router by ZMODEM and by TFTP are shown below in Task E - **Add the CA's public certificate into the PKI Certificate database and set as trusted.**

On the Router

Task D **Create a Security Officer user and enable System Security mode.**

```
Manager > add user=secoff pass=secoff priv=sec
Manager > login secoff
SecOff > set sys name="Certificate VPN demo"
Info (1034003): Operation successful.
SecOff Certificate VPN demo> enable system security
SecOff Certificate VPN demo> set user securedelay=600
```

Task E **Add the CA's public certificate into the PKI Certificate database and set as trusted.**

- As mentioned above in Task C - "Make a copy of certificate file with the file extension expected by the router", the CA's certificate needs to be transferred on the router.
- This step shows a choice of loading the certificate file using ZMODEM or from a TFTP server.

Loading by ZMODEM

```
SecOff Certificate VPN demo> load method=zmodem asyn=0

Router ready to begin ZMODEM file transfers ...
..B0100000023be50
```

On terminal emulator, initiate ZMODEM send of file cacert.cer to the router

```
Info (1048293): ZMODEM, session over.
```

Loading by TFTP

```
SecOff Certificate VPN demo> load fi=cacert.cer serv=10.33.26.11

Info (1048270): File transfer successfully completed.
```

- Add the downloaded file to the PKI Certificate Database

```
SecOff Certificate VPN demo> add pki cert=cacert location=cacert.cer type=ca

Info (1095003): Operation successful.
SecOff Certificate VPN demo> set pki cert=cacert trusted=true

SecOff Certificate VPN demo> sh pki cert

Certificate Database: [ref.#: 14108-1412]

  Name                                  State       MTrust Type    Source
-------------------------------------------------------------------------
  cacert                                TRUSTED     TRUE   CA      COMMAND
-------------------------------------------------------------------------
```

Task E **View the certificate and the fingerprint for confirmation purposes.**

continued

```
SecOff Certificate VPN demo> sh pki cert=cacert

Certificate:
  name ................ cacert
  state ............... TRUSTED
  manually trusted .... TRUE
  type ................ CA
  source .............. COMMAND

  version ............. V3
  serial number .......
  signature alg ....... MD5 with RSA
  public key alg ...... RSA
  not valid before .... 02:21:16 - 21-May-2009  (GMT)
  not valid after ..... 02:21:16 - 05-Oct-2036  (GMT)
  subject ............. cn=CA-Cert, ou=CSG, o=YourCompany,
l=Christchurch, st=Canterbury, c=NZ, dc=alliedtelesis, dc=co, dc=nz
    issuer .............. cn=CA-Cert, ou=CSG, o=YourCompany,
l=Christchurch, st=Canterbury, c=NZ, dc=alliedtelesis, dc=co, dc=nz

  MD5 fingerprint ..... c6a9 d973 cdda c5de a58d 427c bd50 0b72
  SHA1 fingerprint .... 75de e3c5 6693 1d35 6ab1 c5b2 a253 64e3 2bd9
9773
  key fingerprint ..... 90e8 4f1a 2e11 2063 b47e 9c22 b800 73bf e859
5ace

  key usage ........... basic constraints
    subject type ...... CA
    path length ....... No constraint
  subject key ID ...... ba5f4ace3993a8acb23dd6ec96d8bad9b301498c
  authority key ID .... ba5f4ace3993a8acb23dd6ec96d8bad9b301498c

  validation path ..... [ manually trusted, self-signed ]
Source Location:
    file ............. cacert.cer
```

This certificate is valid. The SHA fingerprint of the CA certificate loaded on the router's PKI database matches the fingerprint that we generated on Linux using OpenSSL.

Optionally add the Certificate Revocation List file:

```
load file=ca_crl.crl server=10.33.26.11
add pki crl=ca_crl location=ca_crl.crl
```

Task F **Creating a RSA Public/Private Key Pair**

```
SecOff Certificate VPN demo> create enco key=1 type=rsa length=512
Info (1073278): RSA Key Generation process started.

Info (1073279): RSA Key generation process completed.
SecOff Certificate VPN demo> sh enco key
   ID  Type      Length Digest   Description              Mod   IP
  ------------------------------------------------------------------
    1  RSA-PRIVATE  512 739DAC6B -                        -     -
```

In Task F, we created an RSA public and private key. Now we need to request that the CA sign a certificate to validate the router's ownership of this RSA public/private key pair.

First it is necessary to set the distinguished name on the router to be the same as that used by the CA.

Task G **Create a Certificate Signing Request for the router's own certificate**

```
Certificate VPN demo> set system dist="em=test@solution.net,
cn=router-ATI, ou=CSG_Lab, o=Allied Telesis, l=Christchurch,
st=Canterbury, c=NZ"
```

Then create an enrolment request for keypair 1 (note that in Task F, the ID 'I' was specified when creating the keys).

```
SecOff Certificate VPN demo> create pki
enrollmentrequest=rou_request keypair=1 prot=manual type=pkcs10
format=pem
Info (1095265): PKI Management Request rou_request Completed.
```

The enrolment request is actually an unsigned certificate, that now needs to be sent to the CA to be signed.

```
SecOff Certificate VPN demo> sh fi=*.csr

Filename             Device       Size    Created              Locks
-----------------------------------------------------------------------
rou_request.csr        flash        552     21-May-2009 14:38:58    0
-----------------------------------------------------------------------
```

On the CA

Task H **Receiving the Certificate Signing Request from the router, which it will sign to produce the router's public certificate**

- On the router, upload the signing request by TFTP.

```
SecOff Certificate VPN demo> upload fi=rou_request.csr
serv=10.33.26.11
```

- On the CA, sign the certificate, based in the router's request.

```
[root@multibox URLcerts]# openssl x509 -req -in rou_request.csr -CA
ca/cacert.crt -CAkey ca/cakey.key -CAcreateserial -outform PEM -out
rou_cert.cer -days 9999
Signature ok
subject=/C=NZ/ST=Canterbury/L=Christchurch/O=Allied Telesis/
OU=CSG_Lab/CN=router-ATI/emailAddress=test@solution.net
Getting CA Private Key
Enter pass phrase for ca/cakey.key:

[root@multibox URLcerts]# ls ca/  rou_cert.cer  rou_request.csr
```

- This certificate can now be loaded into the router's PKI Certificate database. First, move the certificate file to the appropriate file transfer directory, ready for download to the router.

```
[root@multibox URLcerts]# cp rou_cert.cer /root
```

On the Router

Task 1 **Add the router's own certificate into the PKI Certificate database, and check it is validated and trusted.**

Now that the certificate has been signed by the CA, it can be loaded back into the router.

```
SecOff Certificate VPN demo> load method=zmodem asyn=0

Router ready to begin ZMODEM file transfers ...
..B0100000023be50
```

In the terminal emulator, initiate the transfer of the signed certificate **rou_cert.cer**

```
Info (1048293): ZMODEM, session over.

SecOff Certificate VPN demo> sh fi=*.cer
Filename            Device      Size    Created             Locks
-------------------------------------------------------------------
cacert.cer    flash         1444     21-May-2009 14:31:05    0
rou_cert.cer  flash         924      21-May-2009 14:46:17    0
-------------------------------------------------------------------
```

Add the file as a self-authenticating certificate. The router can validate that the certificate is signed by the trusted CA:

```
SecOff Certificate VPN demo> add pki cert=rou_cert
location=rou_cert.cer type=self

Info (1095003): Operation successful.
SecOff Certificate VPN demo> sh pki cert

Certificate Database: [ref.#: 27116-1412]
Name                                State      MTrust Type   Source
-------------------------------------------------------------------
  rou_cert                          TRUSTED    FALSE  SELF   COMMAND
  cacert                            TRUSTED    TRUE   CA     COMMAND
-------------------------------------------------------------------
SecOff Certificate VPN demo> sh pki cert=rou_cert

Certificate:
  name ................ rou_cert
  state ............... TRUSTED
  manually trusted .... FALSE
  type ................ SELF
  source .............. COMMAND
  version ............. V1
  serial number ....... 02
  signature alg ....... MD5 with RSA
  public key alg ...... RSA
  not valid before .... 02:38:37 - 21-May-2009  (GMT)
  not valid after ..... 02:38:37 - 05-Oct-2036  (GMT)
  subject ............. em=test@solution.net, cn=router-ATI,
ou=CSG_Lab, o=Alllied Telesis, l=Christchurch, st=Canterbury, c=NZ
  issuer .............. cn=CA-Cert, ou=CSG, o=YourCompany,
l=Christchurch, st=Canterbury, c=NZ, dc=alliedtelesis, dc=co, dc=nz
  MD5 fingerprint ..... 978f 6bc2 fb70 25dd 5534 0e65 b089 f895
  SHA1 fingerprint .... 3043 a19e 8266 3555 c422 9afd 8979 1674 786f
70fd
  key fingerprint ..... 4e04 cf5f 60dc 2359 aac4 dc42 0540 7caa 7400
748c
  key usage ...........
  validation path ..... <- cacert[ manually trusted, self-signed ]
  Source Location:
    file .............. rou_cert.cer
```

On the CA

Task I continued

The router's signed certificate which has now been downloaded and added to the router PKI register can be confirmed as correctly downloaded by comparing its fingerprint with the source certificate file on the Linux CA:

```
[root@multibox URLcerts]# openssl x509 -in rou_cert.cer -SHA1 -
fingerprint|more

SHA1Fingerprint=30:43:A1:9E:82:66:35:55:C4:22:9A:FD:89:79:16:74:78:
6F:70:FD
```

This matches correctly with the SHA1 fingerprint as previously seen on the router's copy of the certificate.

On the Router

Task J **Configuring the network and a VPN access concentrator facility.**

The VLAN configuration:

```
create vlan="private" vid=2
add vlan="2" port=2-5
```

The IP configuration:

```
enable ip
# VLAN1 is the public side
add ip int=vlan1 ip=10.17.90.181 mask=255.255.255.0
# VLAN2 is the private side
add ip int=vlan2 ip=172.28.4.30

add ip rou=0.0.0.0 mask=0.0.0.0 int=vlan1 next=<gateway>
add ip dns prim=x.x.x.x
create ip pool="l2tpclient" ip=172.28.4.31-172.28.4.32
```

Create the user for the L2TP/IPSec VPN remote host:

```
add user=joe pass=friend lo=no telnet=no
```

As an alternative, a RADIUS Server could be configured to handle user authentication requests.

Create the PPP template:

This defines the parameter values that will be set on the dynamic PPP interface that will be attached to the router's end of the L2TP tunnel.

```
create ppp template=1
set ppp template=1 bap=off ippool="l2tpclient" authentication=chap
echo=60 lqr=off
```

The L2TP configuration:

```
enable l2tp
enable l2tp server=both

add l2tp ip=0.0.0.1-255.255.255.254 ppptemplate=1
```

Task J continued

The ISAKMP configuration:

```
create isakmp pol="keys" pe=any enc=3desouter autht=rsasig gro=2
sendd=true sendn=true
```

(You could also, optionally specify the parameters 'setc=true and natt=true on the above command)

```
enable isakmp
```

The IPSec configuration:

Create several different SA specs, so that the policy can support different security association combinations offered by Windows VPN remote host:

```
create ipsec sas=1 key=isakmp prot=esp enc=3desouter hasha=sha
set ipsec sas=1 mod=transport
create ipsec sas=2 key=isakmp prot=esp enc=3desouter hasha=md5
set ipsec sas=2 mod=transport
create ipsec sas=3 key=isakmp prot=esp enc=des hasha=sha
set ipsec sas=3 mod=transport
create ipsec sas=4 key=isakmp prot=esp enc=des hasha=md5
set ipsec sas=4 mod=transport
create ipsec bundle=5 key=isakmp string="1 or 2 or 3 or 4"
```

All policies below define local port, because we expect all tunnels to be externally initiated from clients - i.e.: incoming to this router.

```
The IPSec policy to allow all ISAKMP negotiation and NAT-T traffic
through to their appropriate modules:
create ipsec pol="isakmp" int=vlan1 ac=permit
set ipsec pol="isakmp" lp=500 tra=UDP
create ipsec pol="natt_udp" int=vlan1 ac=permit
set ipsec pol="natt_udp" lp=4500 tra=UDP
```

The IPSec policy for L2TP tunnel:

```
create ipsec pol="l2tpVPN" int=vlan1 ac=ipsec key=isakmp bund=5
peer=ANY
set ipsec pol="l2tpVPN" lp=1701 tra=UDP
```

Add a final policy that allows all other traffic to pass through - so traffic from the LAN behind the router can access the general Internet. If you are allowing direct Internet access, then a firewall needs to be provisioned either on this router or elsewhere on the network:

```
create ipsec pol="internet" int=vlan1 ac=permit
enable ipsec
```

On the Router

Task **Configure the firewall.**

```
enable fire
create fire policy=main
create fire policy=main dy=dynamic
add fire policy=main dy=dynamic user=ANY
add fire policy=main int=vlan1 type=private
```

Dynamic private interfaces are accepted from L2TP, which are from IPSec only.

```
add fire policy=main int=dyn-dynamic type=private
add fire policy=main int=eth0 type=public
```

The firewall allows for internally generated access to the Internet through the following NAT definition.

```
add fire policy=main nat=enhanced int=vlan1 gblinterface=eth0
```

The following NAT definition allows Internet access for remote VPN users by providing address translation.

```
add fire policy=main nat=enhanced int=dyn-dynamic gblinterface=eth0
```

Rules 1 and 2 allow for ISAKMP and the "port floated" IKE/ISAKMP that NAT-T uses.

```
add fire policy=main rule=1 int=eth0 action=allow protocol=udp
ip=<office Internet address> port=500 gblip=<office Internet
address>
gblport=500
add fire policy=main rule=2 int=eth0 action=allow protocol=udp
ip=<office Internet address> port=4500 gblip=<office Internet
address>
gblport=4500
```

Rule 3 becomes the L2TP tunnel allow rule. Additional security is provided by only allowing traffic from IPSec tunnels.

```
add fire policy=main rule=3 int=eth0 action=allow prot=udp
ip=<office Internet address> port=1701 gblip=<office Internet
address>
gblport=1701 encap=ipsec
```

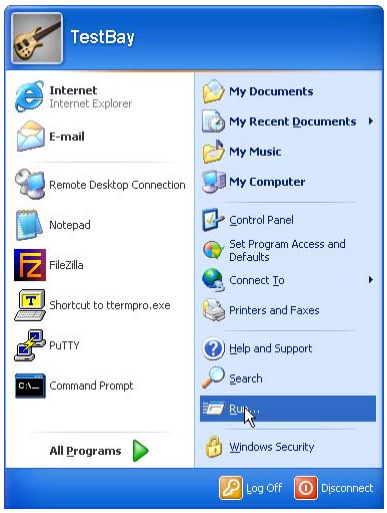
We recommend you use Secure Shell for remote management. Telnet should not be used to a secure gateway.

```
enable ssh server serverkey=2 hostkey=3 expirytime=12
logintimeout=60
add ssh user=secoff password=<secoff password> ipaddress=<trusted
remote ip>
```

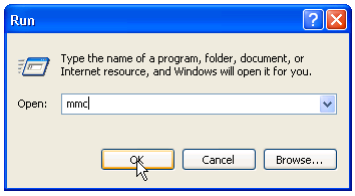
On the VPN remote host

Task L Set-up an MMC Certificates Console to manage your certificates.

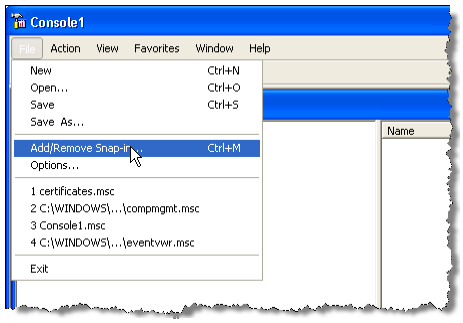
- From the Windows **Start** menu, select **Run...**



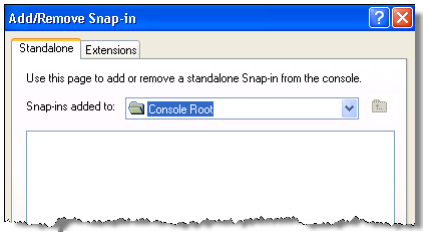
- In the **Run** dialogue box type **MMC**. Click **OK**.



- In the **MMC** Console, select **File** > **Add /Remove Snap-in...**



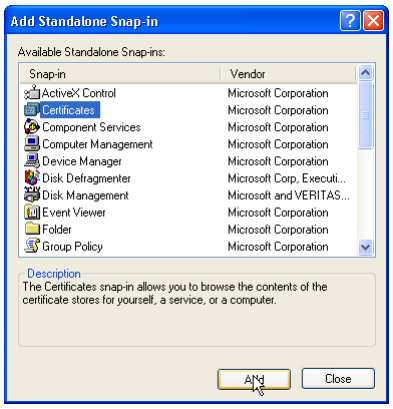
- In the **Add /Remove Snap-in** dialogue box, select **Add...**



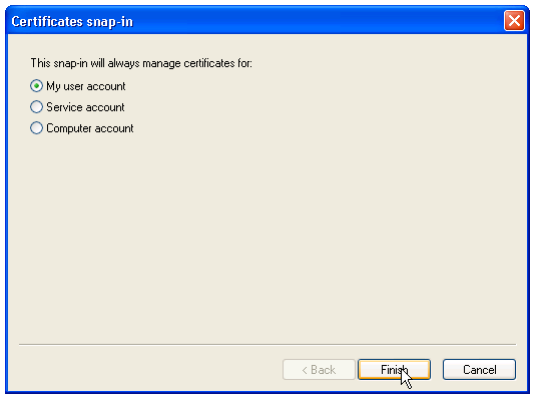
Task L

continued

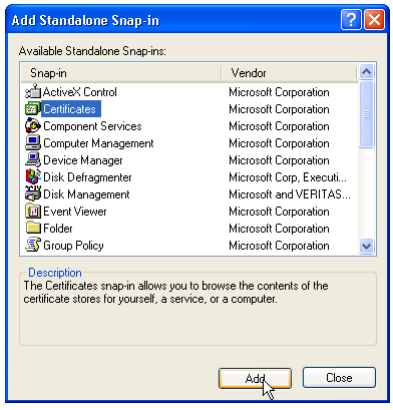
- Select **Certificates** from the list of **Snap-ins**, and click **Add**.



- Select **My user account**, then click **Finish**.
  There are three types of **Certificate Snap-ins** available: User, Service, and Computer. We will add **two** of them.



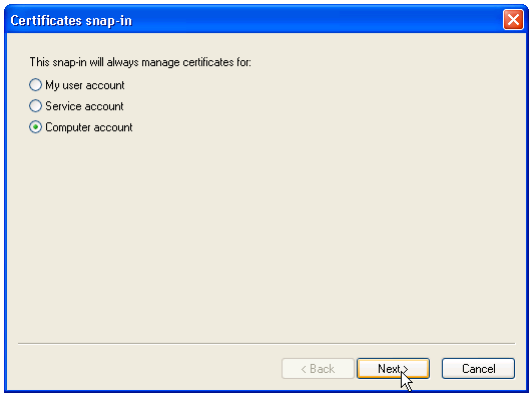
- Back at the **Add Standalone Snap-in** window, select **Certificates** again, and click **Add**. This will allow you to snap-in another Certificate type.



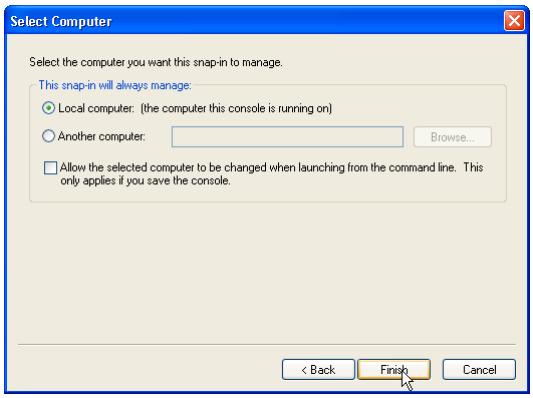
Task L

continued

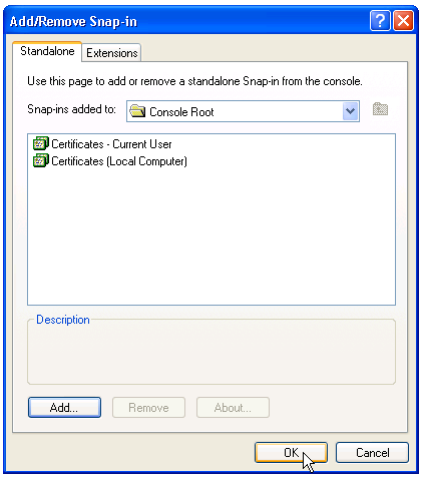
- This time, select **Computer accoun**t, then click **Next**.



- Select **Local computer**, then click **Finish**.



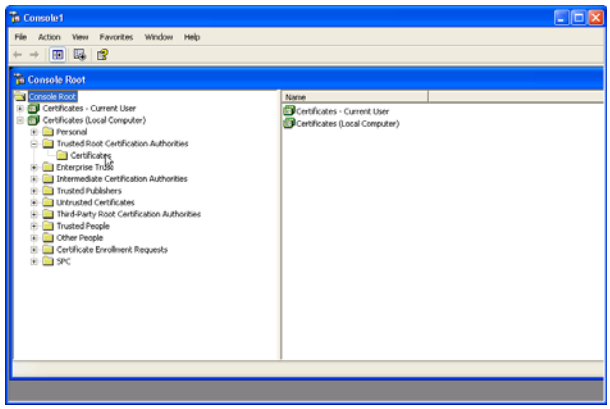
- Confirm that the **Add/Remove Snap-in** window has the **two** certificate types listed, then click **OK**.



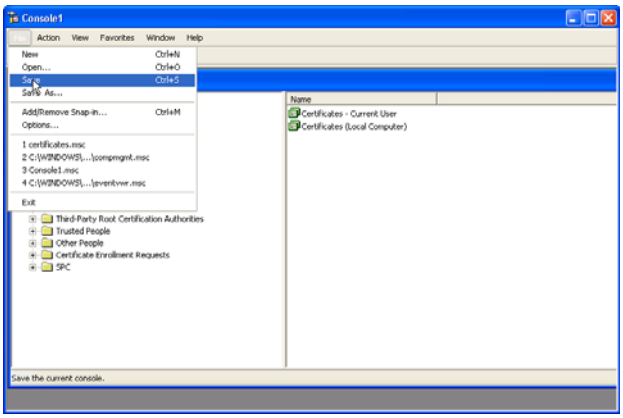
Task L

continued

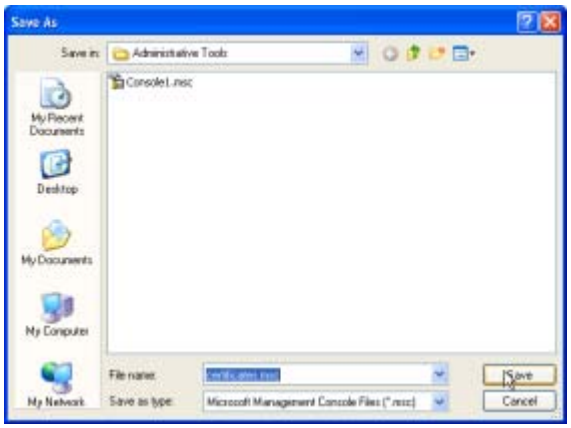
- At the **MMC Console** you should now have the two certificate Snap-ins available. As shown, you can open the hierarchy of certificates.



- To save and create a short-cut to this MMC Console snap-in arrangement, select **File** > **Save**.



- **Save** the msc file under the **Administrative Tools** folder.
  i.e. C:/Documents and Settings/User/Start Menu/Programs/Administrative Tools.

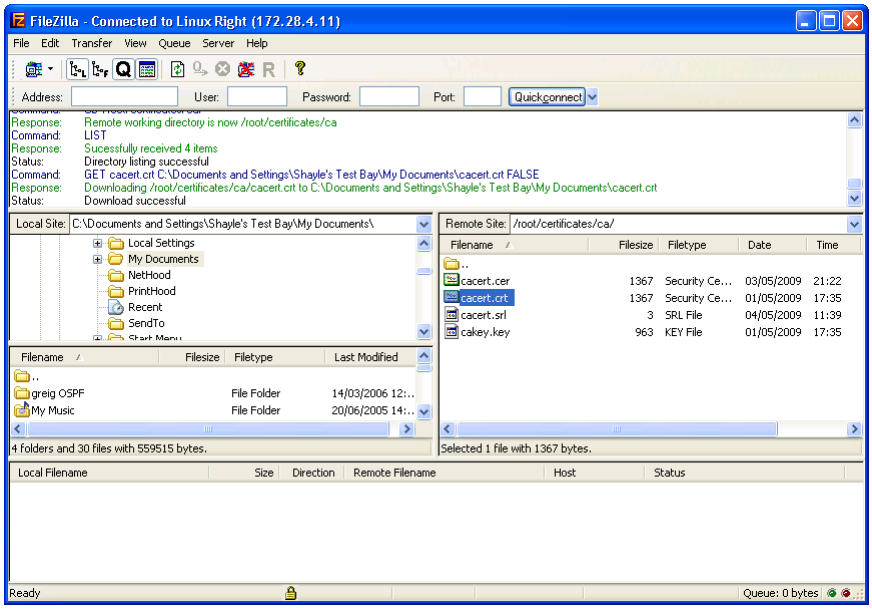


- After saving, your Certificates MMC short-cut should be available from **Start** > **All Programs** > **Administrative Tools**

On the VPN remote host

Task M Download the CA's Certificate to the Windows XP Client.

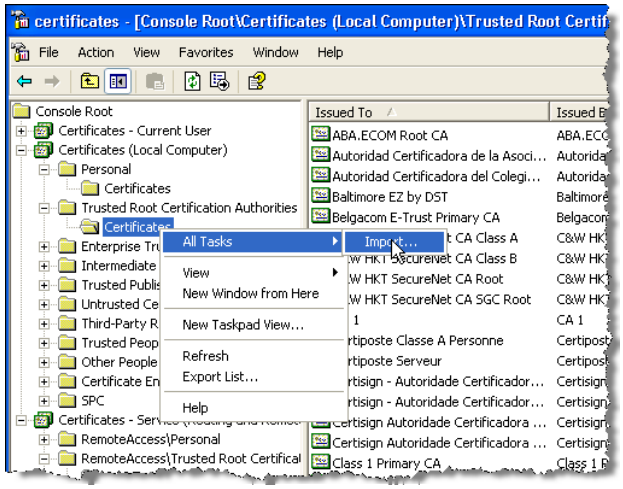
- Use a file transfer method such as FTP. The CA Certificate was created in the earlier section in Task B, "Generating the CA's Public /Private Key Pair and its own public X.509 certificate." on page 5



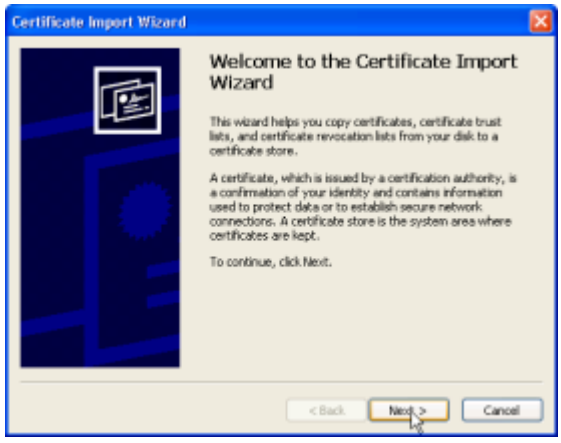
On the VPN remote host

Task N **Import the CA Certificate to the Trusted Root Certification Authority Store.**

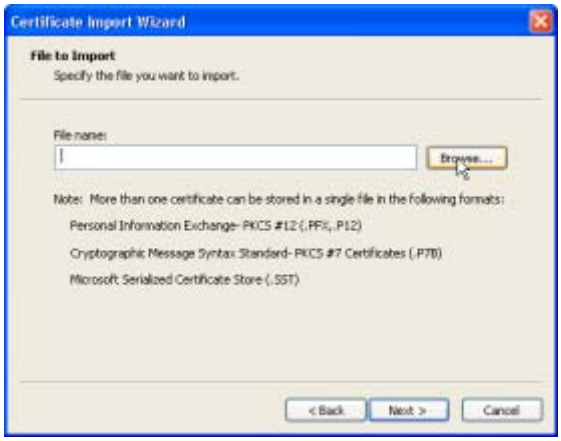
- In the MMC console, select: **Certificates (Local Computer)** > **Trusted Root Certification Authorities** > **Certificates**
- Right-click on **Certificates** and select: **All Tasks** > **Import**...



- This will start the **Certificate Import Wizard**. Click **Next**.



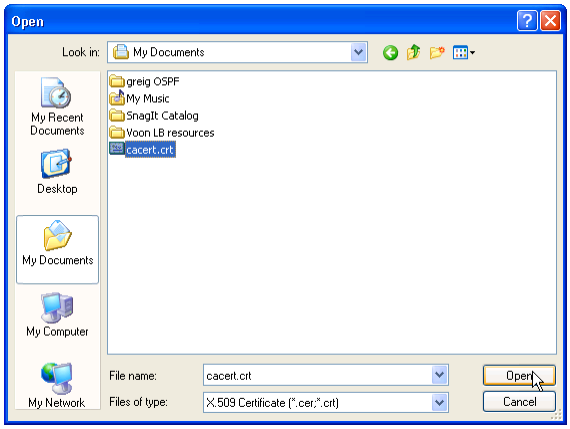
- To import the CA's certificate file, click **Browse...**



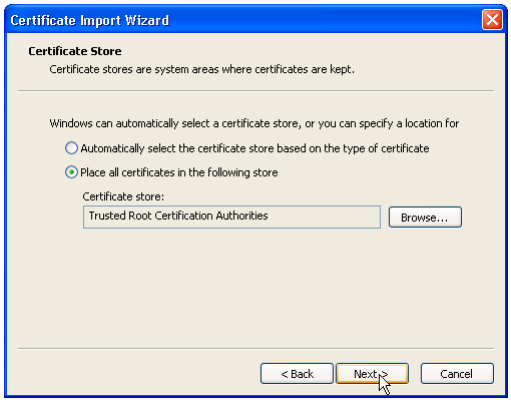
Task N

continued

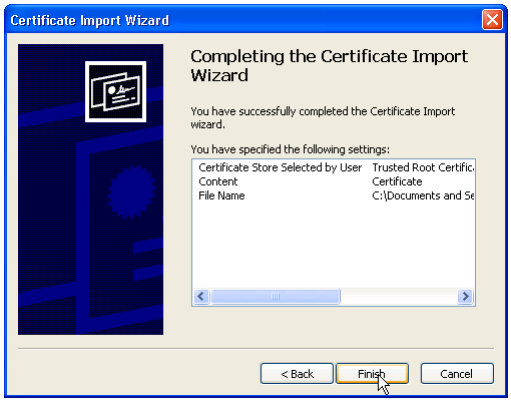
- Select your **cacert** file. Click **Open**.



- The Wizard will automatically prompt a suitable store, in this case **Trusted Root Certification Authorities**. Click **Next**.



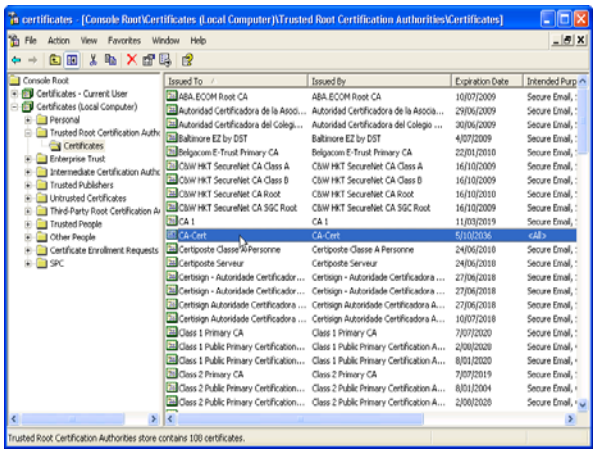
- Confirm the settings you have selected, click **Finish** to end the Wizard, then **OK** to complete.



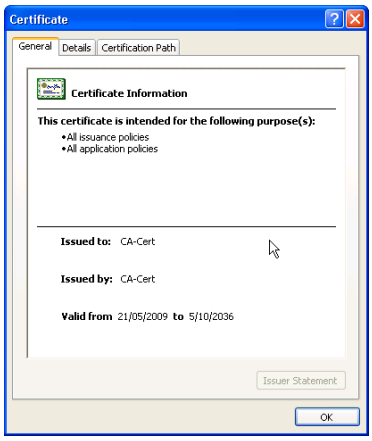
Task N

continued

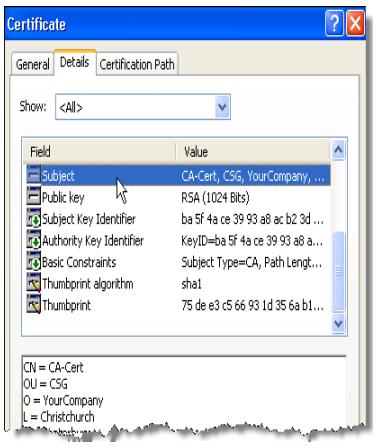
- In the Certificate MMC Console, confirm that the certificate has been imported. The certificates are sorted by the Common Name they were issued to. In this case, the trusted CA certificate for the computer will be found as follows: **Certificates (Local Computer)** > **Trusted Root Certification Authorities** > **Certificates**
  Look in the **Issued To** list for the CA's Common Name.



- Double-click on the certificate to display its detail. It should show as a valid certificate - if it is not valid, an error message will be displayed.



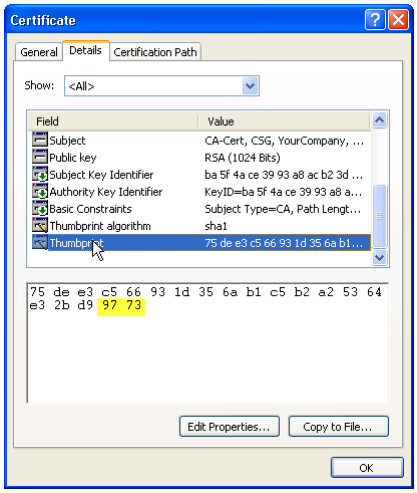
- Verify that the **Subject** (Distinguished Name) details are correct.



Task N

continued

- Verify that the certificate is valid through SHA1 thumbprint. It should be the same value as in Task E (Adding the CA certificate to the router's PKI Database, on page 5), and Task B (Creating the CA's Public X.509 certificate, on page 5). In this example it ends with the numbers: 97 and 73.



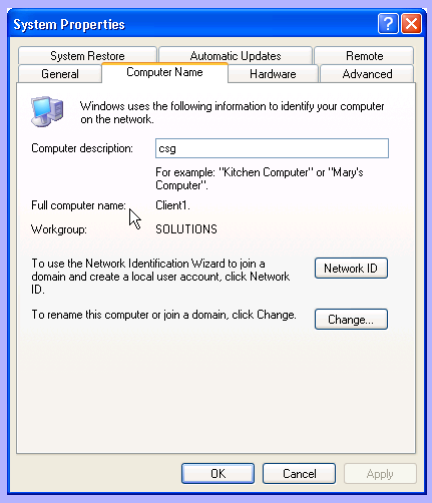
On the CA

Task O Creating a certificate signing request on behalf of the Windows XP VPN remote host machine.

The next steps show the process used on the Linux CA Computer, to make a certificate request on behalf of the Windows XP Client, and then signing that request to produce the certificate.

Preparation points

- Be sure to use the correct common name for the Windows XP Client. The common name must be the computer's name (for example "Client1")



- The OpenSSL command shown in the example below uses the file openssl.cnf. This file is normally installed automatically with an OpenSSL installation. In order for the openssl command to access it, this file may need to be copied into a system PATH directory or to the current directory.

Optional Note:

The file openssl.cnf may be used to specify other certificate fields such as domain name that are not prompted by the openssl command An example of this is shown in Appendix A. on page 52

Example using Linux OpenSSL tool to make Windows XP certificate request

Syntax:

```
openssl req -config openssl.cnf -out name.csr -pubkey -new -keyout
nameKey.pem -outform PEM -nodes
```

Example:

```
[root@localhost certificates]# updatedb
[root@localhost certificates]# locate openssl.cnf
```

Task O continued

This certificate request shows an example of most available Distinguished Name fields being set, these include the Domain Component (DC) fields and the email field.

```
[root@multibox URLcerts]# openssl req -out WindowsXP.csr -pubkey -
new -keyout WindowsXPKey.pem -outform PEM -nodes
Generating a 1024 bit RSA private key
.++++++
.........++++++
writing new private key to 'WindowsXPKey.pem'
-----
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Domain Component 1 (e.g. nz) [nz]:
Domain Component 2 (e.g. co) [co]:
Domain Component 3 (e.g. alliedtelesis) [alliedtelesis]:
Country Name (2 letter code) [NZ]:
State or Province Name (full name) [Canterbury]:
Locality Name (eg, city) [Christchurch]:
Organization Name (eg, company) [YourCompany]:
Organizational Unit Name (eg, section) [OUN]:csg
Common Name (eg, YOUR name) [YourName]:Client1
Email Address []:test@solution.net
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:asdf
An optional company name []:
```

Now you should have a Certificate Request File (WindowsXP.csr) and a private key (WindowsXPKey.pem):

```
[root@multibox URLcerts]# ls
ca/  rou_cert.cer  rou_request.csr  WindowsXP.csr  WindowsXPKey.pem
```

Task P

Produce the Windows XP Public Certificate which the XP computer can import to its Certificate Registry.

Syntax:

```
openssl x509 -req -in name.csr -CA ca/cacert.crt -CAkey ca/cakey.key
-CAcreateserial -outform PEM -out name.pem -days 9999
```

Example:

```
[root@multibox URLcerts]# openssl x509 -req -in WindowsXP.csr -CA ca/
cacert.crt -CAkey ca/cakey.key -CAcreateserial -outform PEM -out
WindowsXP.pem -days 9999
Signature ok
subject=/DC=nz/DC=co/DC=alliedtelesis/C=NZ/ST=Canterbury/
L=Christchurch/O=YourCompany/OU=csg/CN=Client1/
emailAddress=test@solution.net
Getting CA Private Key
Enter pass phrase for ca/cakey.key:
[root@multibox URLcerts]#
```

Task P continued

Now you should have a certificate (*WindowsXP.pem*) and a private key (*WindowsXPKey.pem*):

```
[root@multibox URLcerts]# ls -lct|more
total 24
-rw-r--r-- 1 root root 1099 May 22 12:24 WindowsXP.pem
-rw-r--r-- 1 root root 1106 May 22 12:13 WindowsXP.csr
-rw-r--r-- 1 root root 887 May 22 12:13 WindowsXPKey.pem
drwxr-xr-x 2 root root 4096 May 21 14:38 ca/
-rw-r--r-- 1 root root 924 May 21 14:38 rou_cert.cer
-rw------- 1 root root 552 May 21 14:35 rou_request.csr
```

Verify the Windows certificate sha1 fingerprint:

```
[root@multibox URLcerts]# openssl x509-in WindowsXP.pem-SHA1-
fingerprint|more
SHA1Fingerprint=F8:38:E5:B3:26:13:15:7D:00:80:DC:9B:28:4D:17:23:85:
62:FF:32
```

Task Q **Converting the Windows XP certificate to a form that includes a Private Key.**

We need to combine the certificate and the private key value into a **pkcs12** format file, so that both the certificate and the private key that were created on behalf of Windows XP can be loaded onto Windows XP:

Note that you will be prompted for an export password. You can use any password you like, but make sure you remember this password, as you will need to enter it again when you import the certificate into the Windows PC.

Syntax:

```
openssl pkcs12 -export -in name.pem -inkey nameKey.pem -out name.p12
```

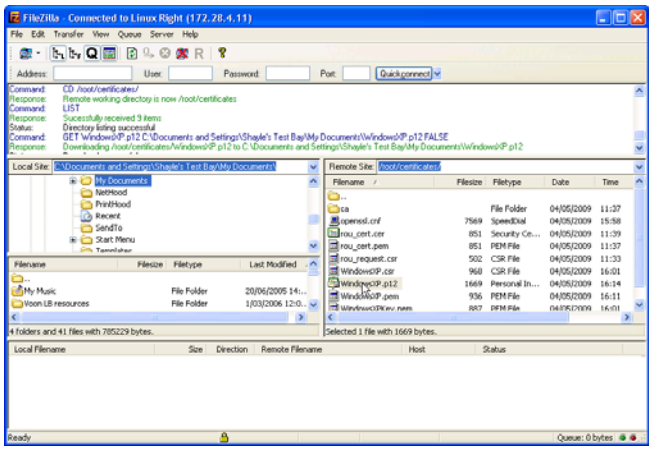
Example:

```
[root@multibox URLcerts]# openssl pkcs12 -export -in WindowsXP.pem -
inkey WindowsXPKey.pem -out WindowsXP.p12
Enter Export Password:
Verifying - Enter Export Password:
[root@multibox URLcerts]# ls *.p12
WindowsXP.p12
```

On the VPN remote host

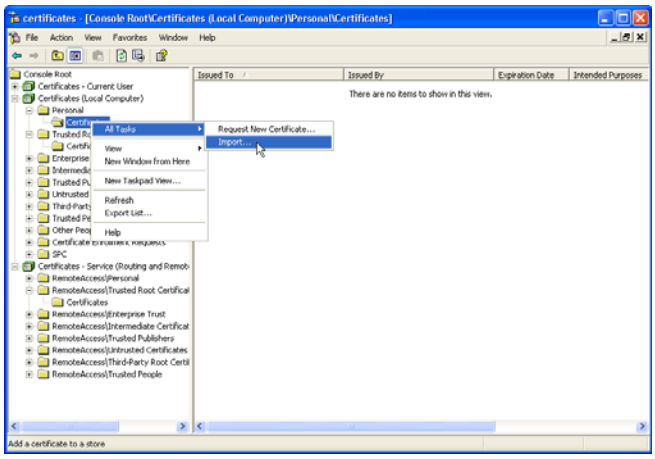
Task R Download the Windows XP Certificate to the Windows XP Client.

- Use a file transfer method such as FTP, to download the Windows XP Client certificate.p12 file.



Import the Windows XP certificate to the Personal Certificate Store

- In the MMC console, select: **Certificates (Local Computer)** > **Personal** > **Certificates**
- Right-click on Certificates and select: **All Tasks** > **Import...**

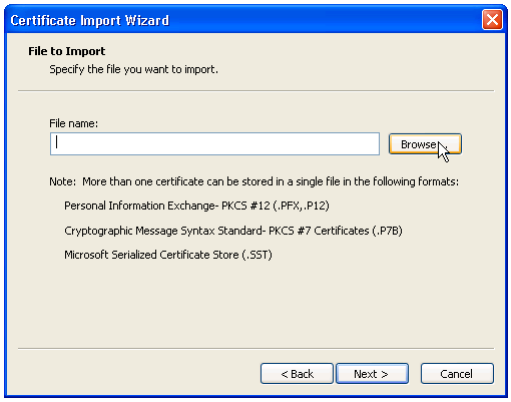


- This will start the **Certificate Import Wizard**.
- Click **Next**.

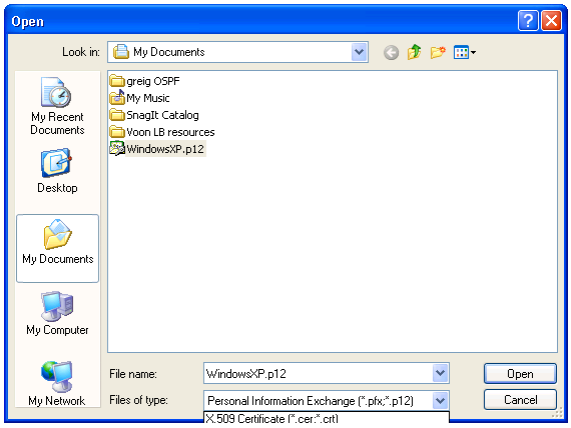
On the VPN remote host

Task R continued

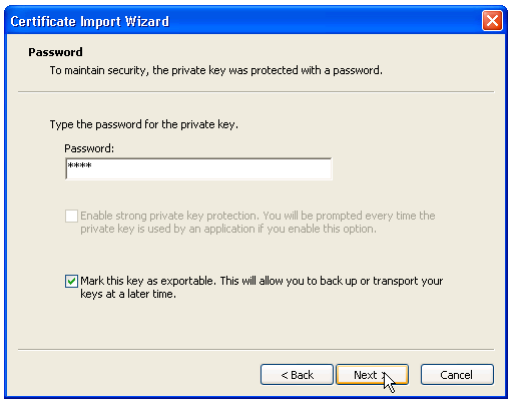
- To import the CA's certificate file, click **Browse**.



- Select your WindowsXP p12 file. Click **Open**.



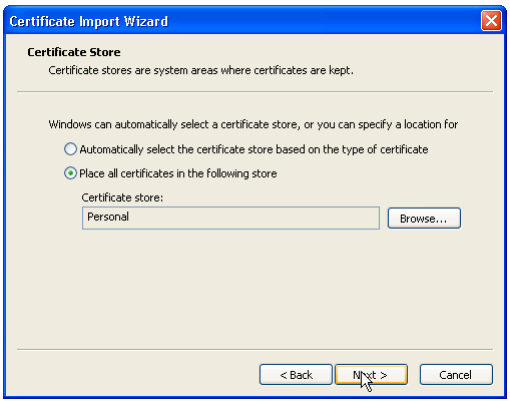
- Type in the **password** of the private key, as used on Linux during the creation of this certificate in Task Q. Click **Next**.



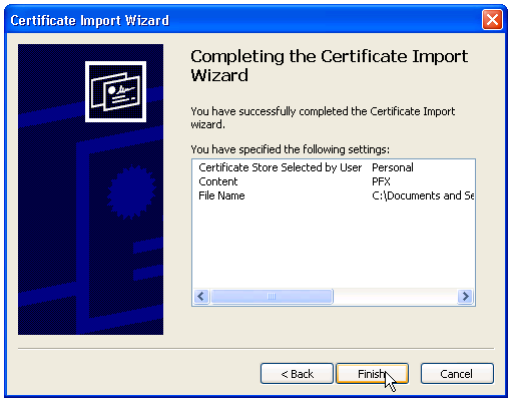
On the VPN remote host

Task R continued

- The Wizard will automatically prompt a suitable store, in this case it is the **Personal** Certificate Store. Click **Next**.



- Confirm the settings you have selected, then click **Finish**.

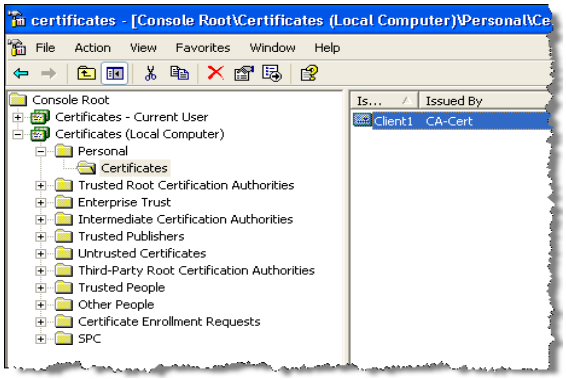


- Click **OK** to complete the import.

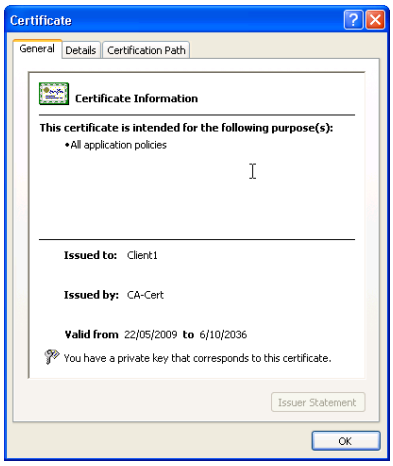
On the VPN remote host

Task R continued

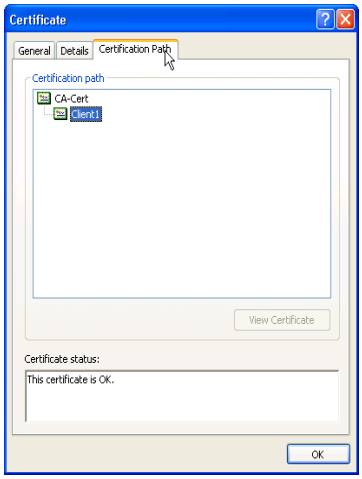
- In the Certificate MMC Console you can confirm that the certificate has been imported. The certificates are sorted by the Common Name they were issued to. In this case, the Personal Certificate for the computer will be found as follows: **Certificates (Local Computer)** > **Personal** > **Certificates.**
- Look in list of certificates for the CA's Common Name.



- Double-click on the certificate and the certificate details will be displayed. It should show as a valid certificate - if it is not valid, an error message will be displayed.



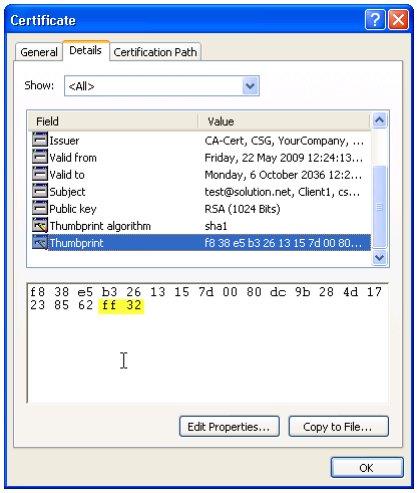
- The **Certification Path** tab, displays which CA issued (signed) this certificate.



On the VPN remote host

Task R continued

- You can also validate the certificate by fingerprint value. It should match the fingerprint seen in Task P. In this example the value ends with **FF 32**:

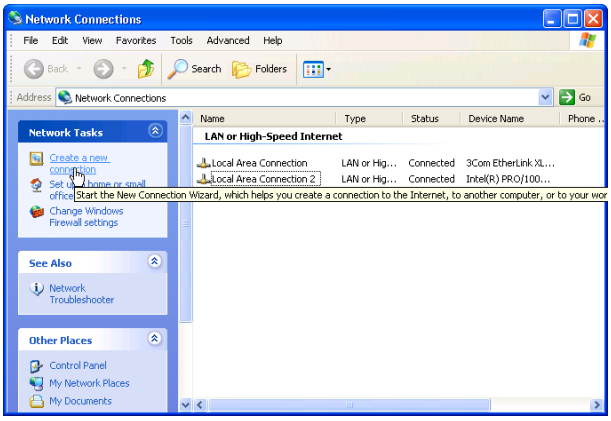


- This certificate will be used for negotiating the VPN link, and will dynamically be added to the router's PKI Database. When that happens you can also verify the fingerprint value of that dynamically added certificate, if desired, as a verification step.

On the VPN remote host

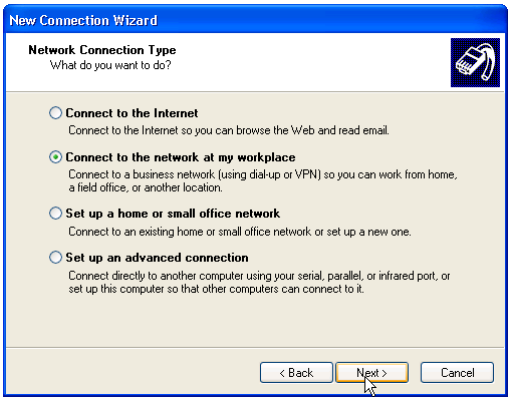
Task S Setting Up the Windows XP Client VPN Network Connection

- Access The Network Connections Summary via **Start** > **Control Panel** > **Network Connections**.
- Select **Create a new connection**.



- The new connection wizard opens, click **Next**.

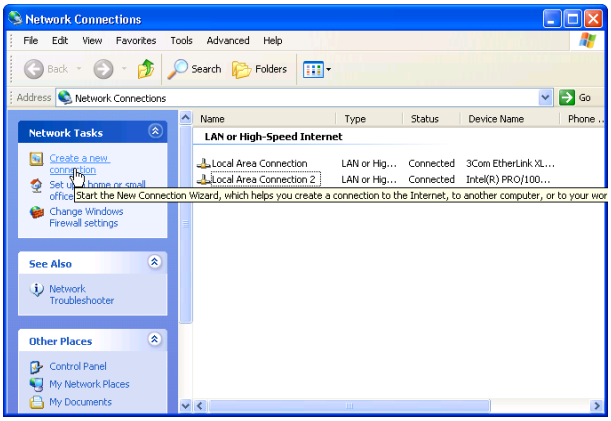
- Select **Connect to the network at my workplace**. This option supports VPN connections.



On the VPN remote host

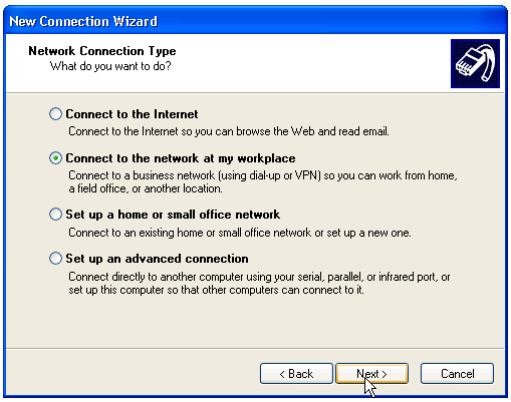
Task S continued

- Select **Virtual Private Network connection**. Click **Next**.



- The new connection wizard opens, click **Next**.

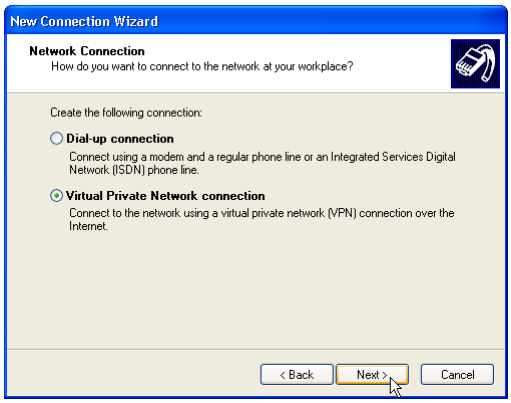
- Select **Connect to the network at my workplace**. This option supports VPN connections.



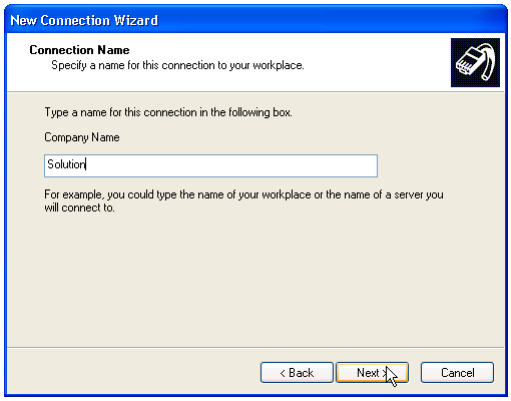
On the VPN remote host

Task S continued

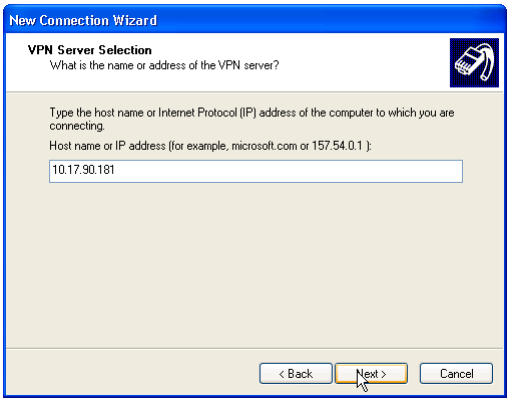
- Select **Virtual Private Network connection.** Click **Next**.



- Type in a **Company Name** for this connection. Click **Next**.



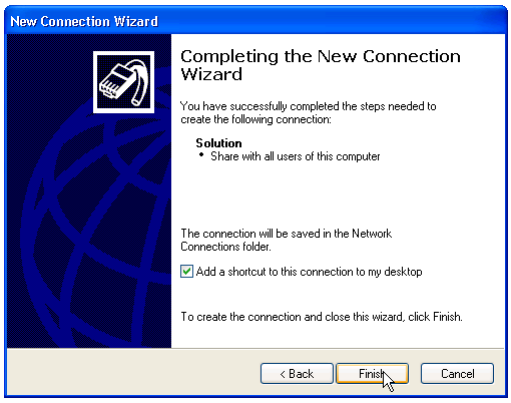
- Type a destination **IP for this VPN** connection. On our VPN router, this will be the address of the public interface that the IPsec policy is applied to. Click **Next**.



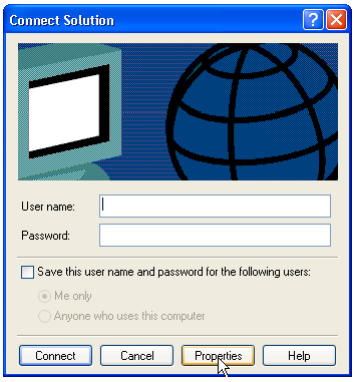
On the VPN remote host

Task S continued

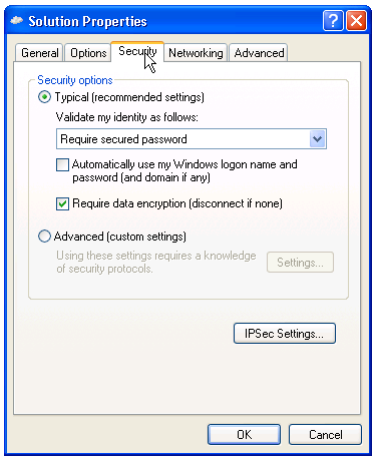
- Complete the New Connection Wizard by clicking **Finish**.



- Other settings need to be adjusted. In the **Connection Solution** dialogue box, click **Properties**.



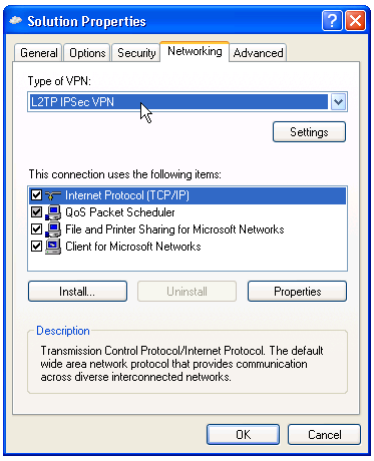
- In the **Security** tab choose **Typical** settings.



On the VPN remote host

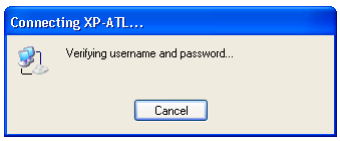
Task S continued

- In the **Networking** tab, choose **L2TP IPsec VPN** from the **Type of VPN** drop-down box. Click **OK**.

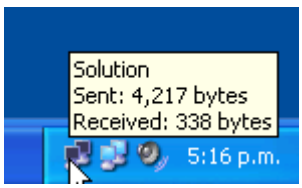


- To test the connection, click **Connect**. For first use, you might choose to enable ISAKMP debugging on the receiving router to track connection progress. Use **enable ISAKMP debug**.

- A **Verifying username and password** dialogue box appears during the connection attempt.



- Once connected, an additional **connection icon** will appear in the system tray. You can double-click the icon for more information.



On the Router

Task T Verification of VPN Connection from Router viewpoint.

On the router, the ISAKMP debug will show as follows for a successful connection.

```
SecOff Certificate VPN demo> enable isakmp debug=all
ISAKMP MAIN exchange 13: New State: IDLE
ISAKMP MAIN: RESP: xchg 13: Started with peer 10.17.90.1
ISAKMP Rx Message
     Cookies:   56a8d065ba62eb36:0000000000000000
     Xchg Type:  IDPROT(2)  Ver: 10  Flags: 00
     MessageID:  00000000   Total Length: 312
     Payload #:  0  Length: 200  Type: Security Association (SA)
       DOI: IPSEC(0)  Situation: 00000001
         Proposal#: 1  Protocol: ISAKMP(1)  #Trans: 5  SPI:
           Transform#: 1
             Transform Id ................. IKE(1)
             Encryption Algorithm.......... 3DESOUTER(5)
             Authentication Algorithm...... SHA(2)
             Authentication Method......... RSA SIGNATURE(3)
             Group Description............. UNKNOWN(14)
             Group Type.................... MODP
             Expiry Seconds................ 28800
           Transform#: 2
             Transform Id ................. IKE(1)
             Encryption Algorithm.......... 3DESOUTER(5)
             Authentication Algorithm...... SHA(2)
             Authentication Method......... RSA SIGNATURE(3)
             Group Description............. 1024(2)
             Group Type.................... MODP
             Expiry Seconds................ 28800
           Transform#: 3
             Transform Id ................. IKE(1)
             Encryption Algorithm.......... 3DESOUTER(5)
             Authentication Algorithm...... MD5(1)
             Authentication Method......... RSA SIGNATURE(3)
             Group Description............. 1024(2)
             Group Type.................... MODP
             Expiry Seconds................ 28800
           Transform#: 4
             Transform Id ................. IKE(1)
             Encryption Algorithm.......... DES(1)
             Authentication Algorithm...... SHA(2)
             Authentication Method......... RSA SIGNATURE(3)
             Group Description............. 768(1)
             Group Type.................... MODP
             Expiry Seconds................ 28800
           Transform#: 5
             Transform Id ................. IKE(1)
             Encryption Algorithm.......... DES(1)
             Authentication Algorithm...... MD5(1)
             Authentication Method......... RSA SIGNATURE(3)
             Group Description............. 768(1)
             Group Type.................... MODP
             Expiry Seconds................ 28800
     Payload #:  1  Length: 24  Type: Vendor ID (VID)
       string=UNKNOWN
       1e 2b 51 69 05 99 1c 7d 7c 96 fc bf b5 87 e4 61 00 00 00 04
     Payload #:  2  Length: 20  Type: Vendor ID (VID)
       string=Microsoft L2TP/IPsec VPN remote host
       40 48 b7 d5 6e bc e8 85 25 e7 de 7f 00 d6 c2 d3
```

On the Router

debug cont.

```
    Payload #:  3  Length: 20  Type: Vendor ID (VID)
         string=draft-ietf-ipsec-nat-t-ike-02\n
         90 cb 80 91 3e bb 69 6e 08 63 81 b5 ec 42 7b 1f
       Payload #:  4  Length: 20  Type: Vendor ID (VID)
         string=UNKNOWN
         26 24 4d 38 ed db 61 b3 17 2a 36 e3 d0 cf b8 19
ISAKMP MAIN: RESP: xchg 13: Rx NAT-T version 2 vendor ID
ISAKMP MAIN exchange 13: New State: SARECV
ISAKMP DOI: IPSEC: Compare transform fail: groupDescription l=2 r=14
ISAKMP MAIN: RESP: xchg 13: Found matching policy = keys
ISAKMP Tx Message
     Cookies:   56a8d065ba62eb36:76d1cff59aa528c8
     Xchg Type:  IDPROT(2)  Ver: 10  Flags: 00
     MessageID:  00000000   Total Length: 80
     Payload #:  0  Length: 52  Type: Security Association (SA)
       DOI: IPSEC(0)  Situation: 00000001
         Proposal#: 1  Protocol: ISAKMP(1)  #Trans: 1  SPI:
           Transform#: 2
             Transform Id ................... IKE(1)
             Encryption Algorithm............ 3DESOUTER(5)
             Authentication Algorithm........ SHA(2)
             Authentication Method.......... RSA SIGNATURE(3)
             Group Description.............. 1024(2)
             Group Type...................... MODP
             Expiry Seconds.................. 28800
ISAKMP MAIN exchange 13: New State: SASENT
ISAKMP Rx Message
     Cookies:   56a8d065ba62eb36:76d1cff59aa528c8
     Xchg Type:  IDPROT(2)  Ver: 10  Flags: 00
     MessageID:  00000000   Total Length: 184
     Payload #:  0  Length: 132  Type: Key Exchange (KE)
       ea 97 76 66 37 64 60 2d ef b0 d3 2a 72 2b 22 3c 94 64 64 26
       90 2d 77 cf e8 be 68 91 8a 5b b9 59 67 ff f3 39 6e fd 5e 39
       c9 d0 73 67 88 25 c0 ee 66 68 b7 96 82 6b 22 d7 8b 3c 92 2c
       b6 b9 8a e6 7c 99 b5 e3 c5 34 a3 b5 dd 80 87 4c 84 e1 52 f0
       f3 61 f4 be 85 53 a3 e6 23 15 5e 0e fc bc 44 0d 7e 0b 52 0f
       37 44 2e 9a c8 e2 a5 fc 8d b5 b4 c3 ca fa 6a 09 c2 6d d7 dd
       be 2c d0 55 87 45 77 50
     Payload #:  1  Length: 24  Type: Nonce (NONCE)
       cb c8 1a 6a dc 96 ec b6 a8 72 2f 9c ca 78 fd f3 a3 d1 22 84
ISAKMP MAIN exchange 13: New State: KERECV
ISAKMP MAIN: RESP: xchg 13: Ni l=20
v=cbc81a6adc96ecb6a8722f9cca78fdf3a3d12284
ISAKMP MAIN: RESP: xchg 13: Nr l=20
v=b9a348d97ff2cffc2c341d1f01b723bf684690b4
ISAKMP MAIN: RESP: xchg 13: COOKIE_I l=8 v=56a8d065ba62eb36
ISAKMP MAIN: RESP: xchg 13: COOKIE_R l=8 v=76d1cff59aa528c8
ISAKMP MAIN: RESP: xchg 13: EncKey l=24
v=7c7645dbf89df549448b7d2dd36a49fb7f93b9
5d68305bc1
ISAKMP MAIN: RESP: xchg 13: IV l=8 v=184fcd78c56a0d50
ISAKMP Tx Message
     Cookies:   56a8d065ba62eb36:76d1cff59aa528c8
     Xchg Type:  IDPROT(2)  Ver: 10  Flags: 00
     MessageID:  00000000   Total Length: 184
```

On the Router

debug cont.

```
Payload #:  0  Length: 132  Type: Key Exchange (KE)
       af 39 83 86 08 b4 38 0b ae c4 f8 d6 b3 7e 24 68 44 de 8f 17
       e4 f3 7d f4 08 7e 2a 7b b4 0e 0b be 16 96 29 d5 de cf 26 37
       cd 36 d8 37 a7 56 a7 59 1d 7e 1f 7b bc 72 1b db e2 70 5b ed
       7e f7 b0 f6 57 c0 6f 30 95 69 51 f4 e3 d4 de 32 d7 8a 44 76
       38 49 2c f6 b5 3c da 4b 43 98 00 ea 03 2e c7 0e d6 5b 64 19
       b3 ce 5a fd bb 7c b8 ee de e1 60 28 09 86 a8 9b 28 63 0d f8
       f5 27 44 6f 81 5d 68 6e
     Payload #:  1  Length: 24  Type: Nonce (NONCE)
       b9 a3 48 d9 7f f2 cf fc 2c 34 1d 1f 01 b7 23 bf 68 46 90 b4
ISAKMP MAIN exchange 13: New State: KESENT

ISAKMP Rx Message (decrypted)
     Cookies:   56a8d065ba62eb36:76d1cff59aa528c8
     Xchg Type:  IDPROT(2)  Ver: 10  Flags: 01
     MessageID:  00000000   Total Length: 1113
     Payload #:  0  Length: 126  Type: Identification (ID)
       Type: DER_ASN1_DN  ProtocolId: 0  Port: 0
      Value: cn=Client1, ou=Sales, o=Allied Telesis, l=Christchurch,
st=
     Payload #:  1  Length: 656  Type: Certificate (C)
       04 30 82 02 87 30 82 01 f0 02 01 03 30 0d 06 09 2a 86 48 86
       f7 0d 01 01 04 05 00 30 81 a3 31 0b 30 09 06 03 55 04 06 13
       02 4e 5a 31 13 30 11 06 03 55 04 08 13 0a 43 61 6e 74 65 72
       62 75 72 79 31 15 30 13 06 03 55 04 07 13 0c 43 68 72 69 73
       74 63 68 75 72 63 68 31 17 30 15 06 03 55 04 0a 13 0e 41 6c
       6c 69 65 64 20 54 65 6c 65 73 69 73 31 19 30 17 06 03 55 04
       0b 13 10 43 75 73 74 6f 6d 65 72 20 53 75 70 70 6f 72 74 31
       0f 30 0d 06 03 55 04 03 13 06 43 53 47 2f 43 41 31 23 30 21
       06 09 2a 86 48 86 f7 0d 01 09 01 16 14 65 78 61 6d 70 6c 65
       40 73 6f 6c 75 74 69 6f 6e 2e 6e 65 74 30 1e 17 0d 30 39 30
       35 30 34 30 34 31 31 33 31 5a 17 0d 33 36 30 39 31 38 30 34
       31 31 33 31 5a 30 74 31 0b 30 09 06 03 55 04 06 13 02 4e 5a
       31 13 30 11 06 03 55 04 08 13 0a 43 61 6e 74 65 72 62 75 72
       79 31 15 30 13 06 03 55 04 07 13 0c 43 68 72 69 73 74 63 68
       75 72 63 68 31 17 30 15 06 03 55 04 0a 13 0e 41 6c 6c 69 65
       64 20 54 65 6c 65 73 69 73 31 0e 30 0c 06 03 55 04 0b 13 05
       53 61 6c 65 73 31 10 30 0e 06 03 55 04 03 13 07 43 6c 69 65
       6e 74 31 30 81 9f 30 0d 06 09 2a 86 48 86 f7 0d 01 01 01 05
       00 03 81 8d 00 30 81 89 02 81 81 00 d7 37 68 b1 e6 df 2a 6a
       93 39 ea a2 c8 72 38 4f ec 79 fa 12 75 12 22 01 ea 80 86 c2
       8a be 57 88 f4 3d 38 d5 b1 d4 21 e4 ee 40 9e 32 eb 1d ed e8
       a5 73 33 0d e6 c7 be 4e 0d e9 c0 a6 5e ac f3 a8 53 13 89 b8
       ea af 91 49 b5 b9 ca d0 75 50 9a 68 07 91 62 43 db 16 7f e8
       9a e3 a9 1f 4a 74 00 d5 5d b6 e5 4b 94 08 7a 40 be c4 a6 41
       a2 7b a6 05 c0 d5 7e 1b 56 e8 cc 7f e1 dc 9e 31 f4 01 b7 f5
       02 03 01 00 01 30 0d 06 09 2a 86 48 86 f7 0d 01 01 04 05 00
       03 81 81 00 84 a2 90 f9 fa 40 9f 51 a1 06 68 fe db 56 5a 93
       b0 22 cc 46 2c c5 39 55 5e 3b 8c 7e ff a7 7f c0 1b 64 75 c1
       d4 85 7e 51 ec b9 b2 0c 55 09 32 da 45 d6 13 c7 83 01 06 6e
       2a c8 72 8f 33 75 a5 64 bc 99 6d 8a eb 46 86 bd 6e 25 26 63
       91 5a e8 fa c7 cf 0a 4c 5f 9d 26 01 22 c5 e2 4b c5 b9 de ed
       90 f0 30 48 31 c5 27 90 b6 47 f2 cd 7f cd 31 6c aa 6e 80 0d
       e7 4d eb 88 d9 01 f2 df d8 40 c0 2e
```

On the Router

debug cont.

```
Payload #:  2  Length: 132  Type: Signature (SIG)
        7a f1 31 f0 e4 be ca 8c 93 48 09 41 59 20 95 e1 96 7c de 67
        72 c8 74 39 70 8b 2e 65 7d 24 15 9c 81 43 a6 21 fd 87 9d e3
        28 1e 58 d8 88 84 0a ab 81 7e fb b8 30 4e a9 28 2d e9 d0 53
        01 49 c4 85 ba 39 56 a2 91 2e 0d da 29 ad ba 2e 8e 37 cd 8f
        86 e9 53 8f c1 fc de 32 a8 99 a8 3a 9a 62 19 00 79 7e 71 28
        33 33 cd 2e 94 a4 0f 0a 35 84 08 96 a4 94 89 06 4d b7 b7 f3
        87 2c 85 7e d8 d9 47 51
      Payload #:  3  Length: 171  Type: Certificate Request (CR)
        04 30 81 a3 31 0b 30 09 06 03 55 04 06 13 02 4e 5a 31 13 30
        11 06 03 55 04 08 13 0a 43 61 6e 74 65 72 62 75 72 79 31 15
        30 13 06 03 55 04 07 13 0c 43 68 72 69 73 74 63 68 75 72 63
        68 31 17 30 15 06 03 55 04 0a 13 0e 41 6c 6c 69 65 64 20 54
        65 6c 65 73 69 73 31 19 30 17 06 03 55 04 0b 13 10 43 75 73
        74 6f 6d 65 72 20 53 75 70 70 6f 72 74 31 0f 30 0d 06 03 55
        04 03 13 06 43 53 47 2f 43 41 31 23 30 21 06 09 2a 86 48 86
        f7 0d 01 09 01 16 14 65 78 61 6d 70 6c 65 40 73 6f 6c 75 74
        69 6f 6e 2e 6e 65 74
ISAKMP MAIN: RESP: xchg 13: Certificate received and validated
ISAKMP MAIN exchange 13: New State: AUTHRECV
ISAKMP MAIN: RESP: xchg 13: RemoteID=DN:cn=Client1, ou=Sales,
o=Allied Telesis,
l=Christchurch, st=
ISAKMP MAIN: RESP: xchg 13: Certificate sent
ISAKMP Tx Message
    Cookies:   56a8d065ba62eb36:76d1cff59aa528c8
    Xchg Type:  IDPROT(2)  Ver: 10  Flags: 00
    MessageID:  00000000   Total Length: 848
    Payload #:  0  Length: 131  Type: Identification (ID)
      Type: DER_ASN1_DN  ProtocolId: 0  Port: 0
      Value: cn=router-ATI, ou=CSG_Lab, o=Allied Telesis,
l=Christchurch, st=
Payload #:  1  Length: 593  Type: Certificate (C)
        04 30 82 02 48 30 82 01 b1 02 01 02 30 0d 06 09 2a 86 48 86
        f7 0d 01 01 04 05 00 30 81 a3 31 0b 30 09 06 03 55 04 06 13
        02 4e 5a 31 13 30 11 06 03 55 04 08 13 0a 43 61 6e 74 65 72
        62 75 72 79 31 15 30 13 06 03 55 04 07 13 0c 43 68 72 69 73
        74 63 68 75 72 63 68 31 17 30 15 06 03 55 04 0a 13 0e 41 6c
        6c 69 65 64 20 54 65 6c 65 73 69 73 31 19 30 17 06 03 55 04
        0b 13 10 43 75 73 74 6f 6d 65 72 20 53 75 70 70 6f 72 74 31
        0f 30 0d 06 03 55 04 03 13 06 43 53 47 2f 43 41 31 23 30 21
        06 09 2a 86 48 86 f7 0d 01 09 01 16 14 65 78 61 6d 70 6c 65
        40 73 6f 6c 75 74 69 6f 6e 2e 6e 65 74 30 1e 17 0d 30 39 30
        35 30 33 32 33 33 39 30 36 5a 17 0d 33 36 30 39 31 37 32 33
        33 39 30 36 5a 30 79 31 0b 30 09 06 03 55 04 06 13 02 4e 5a
        31 13 30 11 06 03 55 04 08 13 0a 43 61 6e 74 65 72 62 75 72
        79 31 15 30 13 06 03 55 04 07 13 0c 43 68 72 69 73 74 63 68
        75 72 63 68 31 17 30 15 06 03 55 04 0a 13 0e 41 6c 6c 69 65
        64 20 54 65 6c 65 73 69 73 31 10 30 0e 06 03 55 04 0b 13 07
        43 53 47 5f 4c 61 62 31 13 30 11 06 03 55 04 03 13 0a 72 6f
        75 74 65 72 2d 41 54 49 30 5c 30 0d 06 09 2a 86 48 86 f7 0d
        01 01 01 05 00 03 4b 00 30 48 02 41 00 dc bf ba ef ef 96 01
        d9 fc ac 80 fe c7 0b f8 24 fd 96 64 b4 96 90 7b 21 98 41 e2
        f1 e9 c0 4d 3a e2 c7 de 43 a2 81 9c cf b5 c6 78 44 56 06 61
        88 42 76 f6 37 8b 4f b1 2e 5e c5 ad f8 7e 51 af 15 02 03 01
        00 01 30 0d 06 09 2a 86 48 86 f7 0d 01 01 04 05 00 03 81 81
```

On the Router

debug cont.

```
00 1a ff ee 51 01 e7 dd 5d 16 dd d3 8d 37 de ac 64 ba 87 c2
       72 ca 23 ce 6c f9 3f 20 88 99 46 65 85 0e 55 a8 ac a7 ad 6b
       31 7f 72 b4 97 fb 4b b8 8b 96 eb fb a5 11 4f 37 98 d6 e7 06
       ac ce 1e 4f 46 79 e3 19 b9 af 98 7e 9c 1f ab a4 ba 3e ef fe
       92 ec f8 40 84 78 ec ac 0d 6a 71 b5 e7 d5 75 97 e1 76 e1 20
       1a 8b 53 53 1a 0a 49 0f ce c9 ad 38 08 ca 0a 37 c3 0c 84 3a
       ae 81 2f 7c 34 80 5d 23 2a
     Payload #:  2  Length: 68  Type: Signature (SIG)
       c7 76 cf 72 cb ba 5d db ee 39 48 a4 7a 63 9f b7 cf d8 14 f4
       8a aa 9d 4a 67 b4 b4 c8 74 ea 46 9c 3e 5d 85 72 6d 6f 35 fd
       47 fd 50 6b 72 09 3c d9 61 e1 d8 07 01 1c e6 74 f5 f3 be f2
       0b a4 d9 29
     Payload #:  3  Length: 28  Type: Notification (N)
       00 00 00 01 01 10 60 02 56 a8 d0 65 ba 62 eb 36 76 d1 cf f5
       9a a5 28 c8
ISAKMP MAIN exchange 13: New State: AUTHSENT

ISAKMP MAIN exchange 13: New State: UP

ISAKMP CORE: Exchange 13 done

ISAKMP QUICK: RESP: xchg 14: Started with peer 10.17.90.1
ISAKMP QUICK exchange 14: New State: WAIT_HASH_SA_NONCE
ISAKMP QUICK: RESP: xchg 14: COOKIE_I l=8 v=56a8d065ba62eb36
ISAKMP QUICK: RESP: xchg 14: COOKIE_R l=8 v=76d1cff59aa528c8
ISAKMP QUICK: RESP: xchg 14: MessageID=03836c7b
ISAKMP QUICK: RESP: xchg 14: IV l=8 v=1fb53490b31bba20
ISAKMP Rx Message (decrypted)
     Cookies:   56a8d065ba62eb36:76d1cff59aa528c8
     Xchg Type:  QUICK(32)  Ver: 10  Flags: 01
     MessageID:  03836c7b   Total Length: 1112
     Payload #:  0  Length: 24  Type: Hash (HASH)
       c2 73 ae c3 09 6c 54 aa 1b b6 c7 6f 3e 19 4a 00 0c 0e 14 b9
     Payload #:  1  Length: 1012  Type: Security Association (SA)
       DOI: IPSEC(0)  Situation: 00000001
         Proposal#: 1  Protocol: ESP(3)  #Trans: 2  SPI: b4736c0e
           Transform#: 1
             Transform Id ................. 3DESOUTER(3)
             Group Description ............ MODP768(1)
             Encapsulation Mode ........... TRANSPORT(2)
             Authentication Algorithm ..... MD5(1)
             Expiry KBytes ................ 250000
             Expiry Seconds ............... 3600
           Transform#: 2
             Transform Id ................. 3DESOUTER(3)
             Group Description ............ MODP768(1)
             Encapsulation Mode ........... TRANSPORT(2)
             Authentication Algorithm ..... SHA(2)
             Expiry KBytes ................ 250000
             Expiry Seconds ............... 3600
```

On the Router

debug cont.

```
Proposal#: 2  Protocol: AH(2)  #Trans: 1  SPI: b4736c0e
  Transform#: 1
    Transform Id ................. SHA(3)
    Group Description ............ MODP768(1)
    Encapsulation Mode ........... TRANSPORT(2)
    Authentication Algorithm ..... SHA(2)
    Expiry KBytes ................ 250000
    Expiry Seconds ............... 3600
Proposal#: 2  Protocol: ESP(3)  #Trans: 1  SPI: 6cbf9d3d
  Transform#: 1
    Transform Id ................. 3DESOUTER(3)
    Group Description ............ MODP768(1)
    Encapsulation Mode ........... TRANSPORT(2)
    Authentication Algorithm ..... NULL(0)
    Expiry KBytes ................ 250000
    Expiry Seconds ............... 3600
Proposal#: 3  Protocol: AH(2)  #Trans: 1  SPI: b4736c0e
  Transform#: 1
    Transform Id ................. MD5(2)
    Group Description ............ MODP768(1)
    Encapsulation Mode ........... TRANSPORT(2)
    Authentication Algorithm ..... MD5(1)
    Expiry KBytes ................ 250000
    Expiry Seconds ............... 3600
Proposal#: 3  Protocol: ESP(3)  #Trans: 1  SPI: 6cbf9d3d
  Transform#: 1
    Transform Id ................. 3DESOUTER(3)
    Group Description ............ MODP768(1)
    Encapsulation Mode ........... TRANSPORT(2)
    Authentication Algorithm ..... NULL(0)
    Expiry KBytes ................ 250000
    Expiry Seconds ............... 3600
Proposal#: 4  Protocol: AH(2)  #Trans: 1  SPI: b4736c0e
  Transform#: 1
    Transform Id ................. SHA(3)
    Group Description ............ MODP768(1)
    Encapsulation Mode ........... TRANSPORT(2)
    Authentication Algorithm ..... SHA(2)
    Expiry KBytes ................ 250000
    Expiry Seconds ............... 3600
Proposal#: 4  Protocol: ESP(3)  #Trans: 1  SPI: 6cbf9d3d
  Transform#: 1
    Transform Id ................. 3DESOUTER(3)
    Group Description ............ MODP768(1)
    Encapsulation Mode ........... TRANSPORT(2)
    Authentication Algorithm ..... SHA(2)
    Expiry KBytes ................ 250000
    Expiry Seconds ............... 3600
Proposal#: 5  Protocol: AH(2)  #Trans: 1  SPI: b4736c0e
  Transform#: 1
    Transform Id ................. MD5(2)
    Group Description ............ MODP768(1)
    Encapsulation Mode ........... TRANSPORT(2)
    Authentication Algorithm ..... MD5(1)
    Expiry KBytes ................ 250000
    Expiry Seconds ............... 3600
```

On the Router

debug cont.

```
            Proposal#: 6  Protocol: ESP(3)  #Trans: 2  SPI: b4736c0e
                    Transform#: 1
                      Transform Id ................. DES(2)
                      Group Description ............ MODP768(1)
                      Encapsulation Mode ........... TRANSPORT(2)
                      Authentication Algorithm ..... MD5(1)
                      Expiry KBytes ................ 250000
                      Expiry Seconds ............... 3600
                    Transform#: 2
                      Transform Id ................. DES(2)
                      Group Description ............ MODP768(1)
                      Encapsulation Mode ........... TRANSPORT(2)
                      Authentication Algorithm ..... SHA(2)
                      Expiry KBytes ................ 250000
                      Expiry Seconds ............... 3600
                  Proposal#: 7  Protocol: AH(2)  #Trans: 1  SPI: b4736c0e
                    Transform#: 1
                      Transform Id ................. SHA(3)
                      Group Description ............ MODP768(1)
                      Encapsulation Mode ........... TRANSPORT(2)
                      Authentication Algorithm ..... SHA(2)
                      Expiry KBytes ................ 250000
                      Expiry Seconds ............... 3600
                  Proposal#: 7  Protocol: ESP(3)  #Trans: 1  SPI: 6cbf9d3d
                    Transform#: 1
                      Transform Id ................. DES(2)
                      Group Description ............ MODP768(1)
                      Encapsulation Mode ........... TRANSPORT(2)
                      Authentication Algorithm ..... NULL(0)
                      Expiry KBytes ................ 250000
                      Expiry Seconds ............... 3600
                  Proposal#: 8  Protocol: AH(2)  #Trans: 1  SPI: b4736c0e
                    Transform#: 1
                      Transform Id ................. MD5(2)
                      Group Description ............ MODP768(1)
                      Encapsulation Mode ........... TRANSPORT(2)
                      Authentication Algorithm ..... MD5(1)
                      Expiry KBytes ................ 250000
                      Expiry Seconds ............... 3600
                  Proposal#: 8  Protocol: ESP(3)  #Trans: 1  SPI: 6cbf9d3d
                    Transform#: 1
                      Transform Id ................. DES(2)
                      Group Description ............ MODP768(1)
                      Encapsulation Mode ........... TRANSPORT(2)
                      Authentication Algorithm ..... NULL(0)
                      Expiry KBytes ................ 250000
                      Expiry Seconds ............... 3600
                     Proposal#: 9  Protocol: AH(2)  #Trans: 1  SPI: b4736c0e
                    Transform#: 1
                      Transform Id ................. SHA(3)
                      Group Description ............ MODP768(1)
                      Encapsulation Mode ........... TRANSPORT(2)
                      Authentication Algorithm ..... SHA(2)
                      Expiry KBytes ................ 250000
                      Expiry Seconds ............... 3600
```

On the Router

debug cont.

```
Proposal#: 9  Protocol: ESP(3)  #Trans: 1  SPI: 6cbf9d3d
            Transform#: 1
              Transform Id ................. DES(2)
              Group Description ............ MODP768(1)
              Encapsulation Mode ........... TRANSPORT(2)
              Authentication Algorithm ..... SHA(2)
              Expiry KBytes ................ 250000
              Expiry Seconds ............... 3600
           Proposal#: 10  Protocol: AH(2)  #Trans: 1  SPI: b4736c0e
            Transform#: 1
              Transform Id ................. MD5(2)
              Group Description ............ MODP768(1)
              Encapsulation Mode ........... TRANSPORT(2)
              Authentication Algorithm ..... MD5(1)
              Expiry KBytes ................ 250000
              Expiry Seconds ............... 3600
           Proposal#: 10  Protocol: ESP(3)  #Trans: 1  SPI: 6cbf9d3d
            Transform#: 1
              Transform Id ................. DES(2)
              Group Description ............ MODP768(1)
              Encapsulation Mode ........... TRANSPORT(2)
              Authentication Algorithm ..... MD5(1)
              Expiry KBytes ................ 250000
              Expiry Seconds ............... 3600
      Payload #:  2  Length: 24  Type: Nonce (NONCE)
        ae 00 08 37 7f ef 4a 65 33 27 2d 7f 3c 3e fe b1 b6 fb 1f cf
      Payload #:  3  Length: 12  Type: Identification (ID)
        Type: IPV4_ADDR  ProtocolId: 17  Port: 1701
        Value: 10.17.90.1
      Payload #:  4  Length: 12  Type: Identification (ID)
        Type: IPV4_ADDR  ProtocolId: 17  Port: 1701
        Value: 10.17.90.181

ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 0: # 1, protid 3,
outspi b4736c0e
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 0 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 1: # 2, protid 2,
outspi b4736c0e
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 1 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 2: # 2, protid 3,
outspi 6cbf9d3d
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 2 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 3: # 3, protid 2,
outspi b4736c0e
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 3 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 4: # 3, protid 3,
outspi 6cbf9d3d
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 4 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 5: # 4, protid 2,
outspi b4736c0e
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 5 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 6: # 4, protid 3,
outspi 6cbf9d3d
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 6 transforms good
```

On the Router

debug cont.

```
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 7: # 5, protid 2,
outspi b4736c0e
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 7 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 8: # 5, protid 3,
outspi 6cbf9d3d
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 8 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 9: # 6, protid 3,
outspi b4736c0e
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 9 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 10: # 7, protid 2,
outspi b4736c0e
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 10 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 11: # 7, protid 3,
outspi 6cbf9d3d
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 11 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 12: # 8, protid 2,
outspi b4736c0e
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 12 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 13: # 8, protid 3,
outspi 6cbf9d3d
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 13 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 14: # 9, protid 2,
outspi b4736c0e
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 14 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 15: # 9, protid 3,
outspi 6cbf9d3d
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 15 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 16: # 10, protid 2,
outspi b4736c0e
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 16 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: rec PROP 17: # 10, protid 3,
outspi 6cbf9d3d
ISAKMP QUICK: RESP: xchg 14: rx msg 1: PROP 17 transforms good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: SA proposals good
ISAKMP QUICK: RESP: xchg 14: rx msg 1: payloads good:
ISAKMP QUICK: RESP: xchg 14: rx msg 1: good
```

On the Router

debug cont.

```
ISAKMP DOI: IPSEC: resp match pol:
   peerIP=10.17.90.1
   filtEnableFlag=00000075
   filtOpaqueFlag=00000000
   selectorsFromPktFlag=00000000
   lAddr=10.17.90.181
   lMask=255.255.255.255
   lAddrLow=0.0.0.0
   lAddrHigh=0.0.0.0
   rAddr=10.17.90.1
   rMask=255.255.255.255
   rAddrLow=0.0.0.0
   rAddrHigh=0.0.0.0
   lPort=1701
   rPort=1701
   lName=
   rName=
   lAddrVer=4
   rAddrVer=4
ISAKMP QUICK: RESP: xchg 14: Match Pol: 2 Local (prot 1) found - 0
ISAKMP QUICK: RESP: xchg 14: Match Pol: 2 Remote (prot 1) found - 0
ISAKMP QUICK: RESP: xchg 14: Match Pol: prop match try: l
00000000000000007d36d
5c r 00000000000000007d2fb9c
ISAKMP QUICK: RESP: xchg 14: Match Pol: matching (prot 2) props 1
ISAKMP QUICK: RESP: xchg 14: Match Pol: (prot 2) tran match try: loc
0 - rem 0
ISAKMP DOI: IPSEC: ATTR match fail: authAlg 2 1
ISAKMP QUICK: RESP: xchg 14: Match Tran: match fail
ISAKMP QUICK: RESP: xchg 14: Match Pol: (prot 2) tran match try: loc
0 - rem 1
ISAKMP QUICK: RESP: xchg 14: Match Tran: match good
ISAKMP QUICK: RESP: xchg 14: Match Pol: matched
ISAKMP QUICK: RESP: xchg 14: proc 1: done good

ISAKMP QUICK exchange 14: New State: SENDING_HASH_SA_NONCE
ISAKMP Tx Message
     Cookies:   56a8d065ba62eb36:76d1cff59aa528c8
     Xchg Type:  QUICK(32)  Ver: 10  Flags: 00
     MessageID:  03836c7b   Total Length: 164
     Payload #:  0  Length: 24  Type: Hash (HASH)
       fc b3 6e 08 fd 5b e6 58 d3 fa 0f 9b ed 71 e6 dd e4 53 1d 1f
     Payload #:  1  Length: 64  Type: Security Association (SA)
       DOI: IPSEC(0)  Situation: 00000001
         Proposal#: 1  Protocol: ESP(3)  #Trans: 1  SPI: 9a764608
           Transform#: 2
             Transform Id ................. 3DESOUTER(3)
             Group Description ............ MODP768(1)
             Encapsulation Mode ........... TRANSPORT(2)
             Authentication Algorithm ..... SHA(2)
             Expiry KBytes ................ 250000
             Expiry Seconds ............... 3600
```

On the Router

debug cont.

```
Payload #:  2  Length: 24  Type: Nonce (NONCE)
       67 32 93 00 b6 3e da 1c e7 ca bf 2d 76 f3 e4 9f 70 45 02 b6
     Payload #:  3  Length: 12  Type: Identification (ID)
       Type: IPV4_ADDR  ProtocolId: 17  Port: 1701
       Value: 10.17.90.1
     Payload #:  4  Length: 12  Type: Identification (ID)
       Type: IPV4_ADDR  ProtocolId: 17  Port: 1701
       Value: 10.17.90.181
ISAKMP Rx Message (decrypted)
     Cookies:   56a8d065ba62eb36:76d1cff59aa528c8
     Xchg Type:  QUICK(32)  Ver: 10  Flags: 01
     MessageID:  03836c7b   Total Length: 52
     Payload #:  0  Length: 24  Type: Hash (HASH)
       10 28 f6 a5 73 07 0b 0c dd 33 6d d1 1e 6a 71 24 ab 88 33 87

ISAKMP QUICK: RESP: xchg 14: rx msg 1: start
ISAKMP QUICK exchange 14: New State: RECEIVING_MESSAGE
ISAKMP QUICK: RESP: xchg 14: rx msg 2: payloads good:
ISAKMP QUICK: RESP: xchg 14: rx msg 2: good
ISAKMP CORE: Exchange 14 done

ISAKMP QUICK exchange 14: New State: DONE
```

On the Router

Task T **Other confirmation commands**.

Here are commands to verify that both the ISAKMP and IPsec Security Associations of the VPN are established, that the PPP link crossing the VPN is open, and that a host specific route to the VPN remote host has been added over the VPN PPP link:

```
SecOff Certificate VPN demo> sh isa sa

                      Expiry Limits - hard/soft/used
SA Id PeerAddress     EncA.  HashA. Bytes                Seconds
-------------------------------------------------------------------
4     10.17.90.1      3DES   SHA    -/-/-              28800/27360/68

SecOff Certificate VPN demo> sh ipsec sa

SA Id  Policy              Bundle  State    Protocol    OutSPI
InSPI
-------------------------------------------------------------------
   3  l2tpVPN               5  Valid    ESP    3027463182  2591442440
SecOff Certificate VPN demo> sh ppp

  Name          Enabled  ifIndex  Over              CP          State
-------------------------------------------------------------------
 *ppp0            YES     10                      IPCP        OPENED
                                 tnl-38326         LCP         OPENED
-------------------------------------------------------------------

SecOff Certificate VPN demo> sh ip rou

IP Routes
-------------------------------------------------------------------
Destination     Mask          NextHop        Flags  Interface
Age
DLCI/Circ.      Type    Policy  Protocol   Tag       Metrics
Pref
-------------------------------------------------------------------
0.0.0.0         0.0.0.0         172.28.0.1      -----  vlan2
169447
-             direct   0      static     -         1             360
10.17.90.0      255.255.255.0    0.0.0.0        -----  vlan1
169447
-             direct   0      interface  -         1               0
172.28.0.0      255.255.0.0      0.0.0.0        -----  vlan2
169447
-             direct   0      interface  -         1               0
172.28.4.31     255.255.255.255  0.0.0.0       -----  ppp0
78
-             direct   0      interface  -         1               0
-------------------------------------------------------------------
```

Caveat statement

This How To document uses Allied Telesis routers, Windows XP, and the OpenSSL Open Source toolkit (also called OpenSSL tool), to demonstrate an X.509 Certificate VPN solution. The certificates used contain many fields of information, and to achieve this the solution also shows how to modify the OpenSSL tool's configuration file to support the Domain Component (DC) fields.

Usually certificates do not utilise all the fields of information that could be used in a certificate, but if you intend to use many fields please be aware of the following limitation.

OpenSSL tool limitation

Because of a suspected limitation of the OpenSSL tool we need to limit the length of the Certificate Signing Request file that the router creates. The string that you configure as **System Distinguished Name** needs to be limited to a string less than around **136** characters. This will ensure we create a Certificate Signing Request (CSR) file that an OpenSSL tool, acting as Certificate Authority, can sign.

Note that 136 characters is an ample amount for most certificate requirements.

Appendix

OpenSSL configuration file - adjustment to support additional fields

This How To document uses the OpenSSL tool to create certificates which contain many fields of information, including the additional Domain Component or DC fields. In order to support the use of the additional fields, the OpenSSL configuration file (.cnf) may need some changes. This appendix provides an extract from a modified configuration file:

[root@localhost URL_certs]# more /usr/local/ssl/openssl.cnf

... Search forward in the configuration file to find this section of the document, and alter as per example below...

```
< cut >
# For the CA policy
[ policy_match ]
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
domainComponent         = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
domainComponent         = optional

####################################################################
[ req ]
default_bits            = 1024
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes
x509_extensions = v3_ca # The extentions to add to the self signed cert

# Passwords for private keys if not present they will be prompted for
# input_password = secret
# output_password = secret

# This sets a mask for permitted string types. There are several options.
# default: PrintableString, T61String, BMPString.
# pkix   : PrintableString, BMPString.
# utf8only: only UTF8Strings.
# nombstr : PrintableString, T61String (no BMPStrings or UTF8Strings).
# MASK:XXXX a literal mask value.
# WARNING: current versions of Netscape crash on BMPStrings or UTF8Strings
```

```
# so use this option with caution!
string_mask = nombstr

# req_extensions = v3_req # The extensions to add to a certificate request

[ req_distinguished_name ]
0.domainComponent               = Domain Component 1 (e.g. nz)
0.domainComponent_default       = nz

1.domainComponent               = Domain Component 2 (e.g. co)
1.domainComponent_default       = co

2.domainComponent               = Domain Component 3 (e.g. alliedtelesis)
2.domainComponent_default       = alliedtelesis

countryName                     = Country Name (2 letter code)
countryName_default             = NZ
countryName_min                 = 2
countryName_max                 = 2

stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = Canterbury

localityName                    = Locality Name (eg, city)
localityName_default            = Christchurch

0.organizationName              = Organization Name (eg, company)
0.organizationName_default      = Your Organization

# we can do this but it is not needed normally :-)
#1.organizationName             = Second Organization Name (eg, company)
#1.organizationName_default     = World Wide Web Pty Ltd

organizationalUnitName          = Organizational Unit Name (eg, section)
organizationalUnitName_default  = SW AW Sustaining

commonName                      = Common Name (eg, YOUR name)
commonName_default              = Your Name
commonName_max                  = 64

emailAddress                    = Email Address
emailAddress_max                = 64

# SET-ex3                       = SET extension number 3

[ req_attributes ]
challengePassword               = A challenge password
challengePassword_min           = 4
challengePassword_max           = 20

unstructuredName                = An optional company name

<cut>
```

USA Headquarters | 19800 North Creek Parkway | Suite 100 | Bothell | WA 98011 | USA | T: +1 800 424 4284 | F: +1 425 481 3895
European Headquarters | Via Motta 24 | 6830 Chiasso | Switzerland | T: +41 91 69769.00 | F: +41 91 69769.11
Asia-Pacific Headquarters | 11 Tai Seng Link | Singapore | 534182 | T: +65 6383 3832 | F: +65 6383 3830
www.alliedtelesis.com

C613-16144-00 REV A