Virtual Private Networks using IPsec-tools

Important note: this document is not yet finished – several sections still need transforming from their present note form to something more readily digestible!

Overview

Why you might want a VPN

A Virtual Private Network (VPN) creates, as its name suggests, a private interconnect between two endpoints using existing infrastructure. It is described as virtual because you need not, and probably won't in most cases, own or even know what network elements exist between the two endpoints.

So why is this useful, or even desirable?

It allows you to create a secure extension of a network in order that one or more chosen computers can become temporary members of that same network. The inclusion of suitable encryption ensures that the traffic flowing over the virtual network remains private irrespective of whether or not you own and therefore control all of the network elements through which the traffic is transferred.

In more practical terms it means that you can connect into your home network from wherever you are in the world without anyone else being able to intercept the traffic. It also means that you can temporarily extend your network into a friends' in order to complete your weekly off-site backup!

What a VPN needs to offer

An ideal VPN needs to provide the following important characteristics:

Authentication

This security measure is designed to establish the validity of a transmission, message, or originator, or provide a means of verifying an individual's authorisation to receive specific categories of information.

Integrity

This measure is designed to enable the recipient of a packet of data to determine whether it has been modified whilst in transit (e.g. by a hostile node that the data passes through en route to its destination) or whether it is untouched since it was transmitted by the originator. It is also desirable to be able to optionally detect, and therefore prevent, the replay of a series of packets sent earlier.

Confidentiality

This characteristic provides the ability to stop an intercepting party from viewing the plain-text contents of the transmission even if they have managed to capture the raw packets. It is synonymous with encryption.

IPsec

Having decided on the desirability of a VPN then Linux, as usual, provides several possible implementation options including OpenVPN, OpenSwan and IPsec-tools. IPsec provides considerable opportunities for inter-operability with other implementations (e.g. Cisco VPN routers, Apple iPhone) and therefore this article concentrates solely on IPsec and specifically the IPsec-tools [] implementation.

IPsec is specified as a series of IETF RFCs, beginning with RFC4301. []

At the link layer most computer-generated traffic transported within modern networks uses Ethernet as its encapsulation. Whilst not the basis of this article it is important to realise that most of the commonly used applications transport their traffic as a series of layered protocols, as illustrated in Figure XX below. IP forms the network layer protocol and therefore virtually all traffic of interest (e.g. HTTP, POP/SMTP, and RSYNC) can be transparently transported in an IPsec link.

Ethernet ARP IP UDP TCP

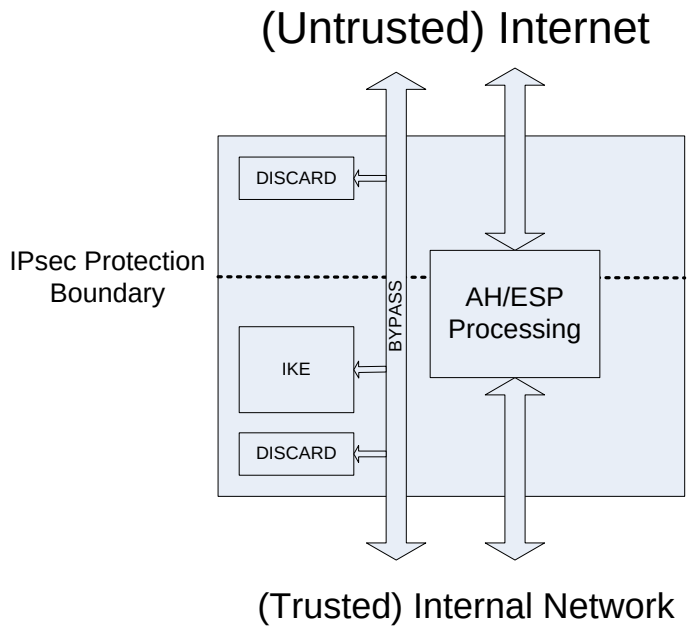
Application layer - DHCP, DNS, HTTP

Transport layer - TCP, UDP

Network layer (Internet layer) - IP, IGMP, ICMP, ARP

Link layer (Network Interface layer) - Ethernet

One type of traffic which cannot be transported without additional effort is multicast traffic since the indicated IP destination address will not normally indicate that the traffic should be prepared for transmission within the IPsec link. Encapsulation protocols, such as GRE, can be used to encapsulate multicast traffic before it traverses an IPsec link, but it should be noted that this requires an additional 24 bytes per packet which must still fit inside the MTU of the link.



IPsec is comprised of multiple protocols and procedures which:

1. Determine which traffic is to be transported within a VPN, e.g. on the basis of its IP destination address or its destination UDP port. This is determined by the defined Security Policies, which are described later in this document.

2. Optionally negotiate the algorithms to be used by the end-points when encrypting/decrypting the traffic. This is achieved using the IKE protocol.

3. Describe the modification of the original IP packets into encapsulated IPsec packets (AH and ESP) which will be transferred over the network. This is achieved by the Linux kernel.

These protocols and procedures are considered further in the following sections.

Tunnel vs. Transport mode

IPsec can be operated in either tunnel mode or transport mode. Each of these modes has its own particular uses and care should be taken to ensure that the correct one is selected:

- *Tunnel mode* is most commonly used between VPN gateways, or at an end-station to a gateway, the gateway acting as a proxy for one or more hosts behind it.
- *Transport mode* is used between end-stations or most commonly between an end-station and a gateway

Transport mode requires all end-station devices to be IPsec aware and is therefore usually used for multiple “roadwarrior” laptops where each machine will have a copy of the VPN client software installed and configured.

Tunnel mode is usually used for semi-permanent site-to-site scenarios since it removes the need for individual machines to be IPsec aware.

The two scenarios (roadwarrior and site-to-site) mentioned above are discussed in further detail later in the article.

It is important to note that the original IP traffic is re-formatted differently when utilising the two traversal modes and this is illustrated later in the document when discussing IPsec encapsulation types.

Internet Key Exchange (IKE) protocol

IKE is described in a series of RFCs including RFC 2407, 2408, 2409; 4306. Its purpose is to negotiate and maintain a supportable security association (SA) with a peer. This is necessary because there are many encryption algorithms, hashing algorithms and related configuration parameters which are necessary to be configured for a VPN gateway. It is by no means certain that the gateways will be able to find an agreeable set which they both support in order to establish a bidirectional link with each other. It is possible to manually configure the peers but given the very large number of configurable parameters this is very error prone and much more difficult to manage.

It should also be noted that the negotiated SAs may have lifetimes applied and consequently ongoing renegotiation between the peers may be required to replace expired SAs.

The IPsec-utils package provides an IKE v1 compatible daemon, called racoon, which handles the negotiations between peers. IKE traffic between the negotiating peers is transferred using the UDP protocol on destination port 500. Suitable firewall rules must be in place in both the modem and Linux hosts for this traffic to traverse successfully.

The negotiation process is performed in two distinct phases during which specific information is requested and transported:

Phase 1

During this phase the peers aim to authenticate each other using one of the supported means (e.g. pre-shared secret keys, Extended Authentication or RSA certificates). The peers will negotiate the security parameters to be used and then establish shared secrets which are used to seed the encryption processes at each peer. The generation of shared secrets is achieved using the Diffie Hellman algorithm and is detailed in [].

Within phase 1 two distinct negotiation modes are supported.

> **Main mode:**
>
> Main mode requires the transfer of 6 messages between the negotiating peers. Security Association negotiation is achieved with just one message sent from each peer. Diffie Hellman exchange requires a further message in each direction and finally authentication of the peers is completed with the final two messages.
>
> **Aggressive mode:**
>
> The alternative, aggressive mode, requires the transfer of only 3 messages, which is obviously quicker but relies on the Diffie Hellmann group being previously agreed. It also includes no cookies with which to prevent DoS type attacks. This mode is often used in road warrior scenarios where it allows faster establishment of the VPN.

Phase 2

Phase 2 begins once phase 1 has been successfully completed, and all messages between the peers are assured to be confidential using the SA negotiated in phase 1 *[was main mode SA]*. Phase 2 is used to determine the SAs to be used for the IPsec traffic. Note that one SA exists for each direction

of the link. Phase 2 prevents replay attacks by using nonces. The final stage of phase 2 generates the keys to be used for the IPsec SAs.

In the IPsec-tools implementation the racoon daemon provides the IKE daemon functionality.

Authentication methods

General overview – better moved into IKE section?

Binds a public key to an identity (IP, FQDN, ASN1DN)

- Pre-shared Keys
- Certificates
- RSA PSK Extended authentication (XAuth) – road-warrior

PKI

RSA

X509

Security Associations

Security Associations are stored within an indexable database in each peer to allow the simple referencing of a series of security related parameters. A Security Association is only applicable to traffic in a single direction between the peers – consequently for bidirectional traffic flow two SAs must be established. The security related parameters which the SA provides are required for the generation and decoding of traffic traversing between the peers. The SPI field transferred in both AH and ESP protocols directly indexes the SA database.

For each security association the following parameters are stored:

- ESP or AH, ie which protocol type is used
- Tunnel or transport mode
- Encryption algorithm
- Authentication algorithm
- Shared session keys
- Lifetime (time, data volume)

Security Policies

The Security policies define which traffic is subject to transfer within an IPsec protected link. Again the policies are stored in a database and specify filters to determine which traffic is subject to the policy and then an action to apply to traffic matching the filter.

The filters can specify matching source and destination IP addresses (or Fully Qualified Domain Names) as well as UDP/TCP destination ports.

The actions which are applicable on detection of traffic matching the particular filter includes:

- Discard (the traffic is actively discarded)
- None (the traffic is transferred without IPsec security applied)
- IPsec (the traffic has IPsec security applied)

For traffic subject to an IPsec action then it additionally defines all the parameters which apply to the traffic, some of which include:

- Protocol (AH, ESP, IPCOMP, TCP)
- Mode (Transport or tunnel)
- Source and Destination endpoints (their IP addresses)
- Level (Default, Use, Require, Unique)

Linux Kernel instructions:

- Default means the kernel consults the system wide default for the protocol you specified, e.g. the esp_trans_deflev sysctl variable, when the kernel processes the packet.

- Use means that the kernel uses an SA if it's available; otherwise the kernel keeps normal operation.

- Require means an SA is required whenever the kernel sends a packet matched with the policy.

- Unique is the same as require; but in addition, it allows the policy to match the unique out-bound SA. You just specify the policy level unique, racoon will configure the SA for the policy.

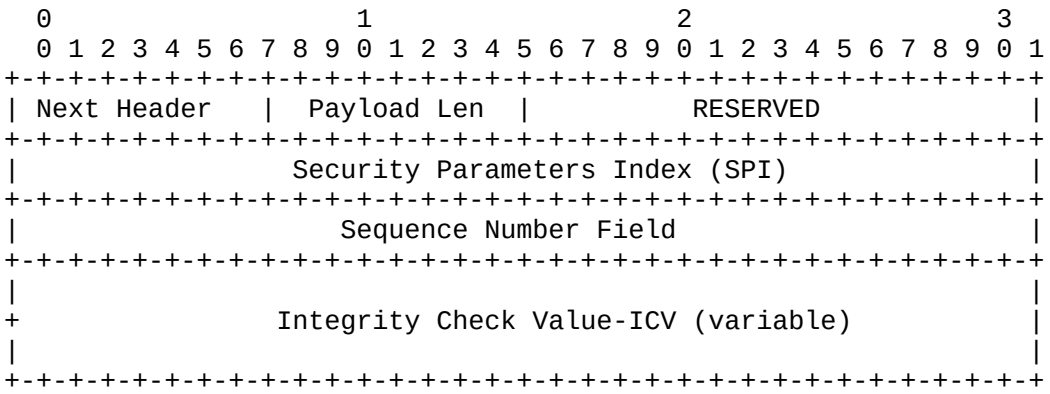
IPsec Traffic Protocols

IPsec traffic is transferred in one of two protocol types. These are called Authentication Headers (AH) and Encapsulating Security Payloads (ESP) and are described below.

AH is defined to be protocol 51 and can be selected for filtering within iptables using the ‘-p ah’ construct whereas ESP is protocol 50 and can be selected using ‘-p esp’.

AH

Authentication Headers satisfy only two of the desired characteristics for a VPN that were described above. They support authentication and integrity verification only. Whilst it is useful to know that the traffic comes from a trusted source and that the data has not been tampered with, the fact that the confidentiality of the traffic can not be guaranteed means that the protocol is of limited value for most purposes. AH traffic is specified in RFC4302 and the format of the packet is shown in the figure below:



Fields of particular interest include the SPI, Sequence number and ICV. The SPI field transfers an index which tells the end point which previously agreed algorithms are to be applied when it receives the packet.

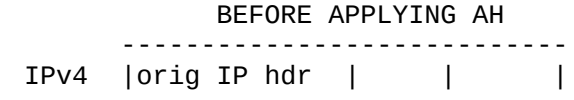
The sequence number field is incremented for each packet of traffic transferred in a given direction between two end-points. Its aim is to prevent replay attacks since an endpoint can track the received sequence number and choose to drop packets which it has received before.

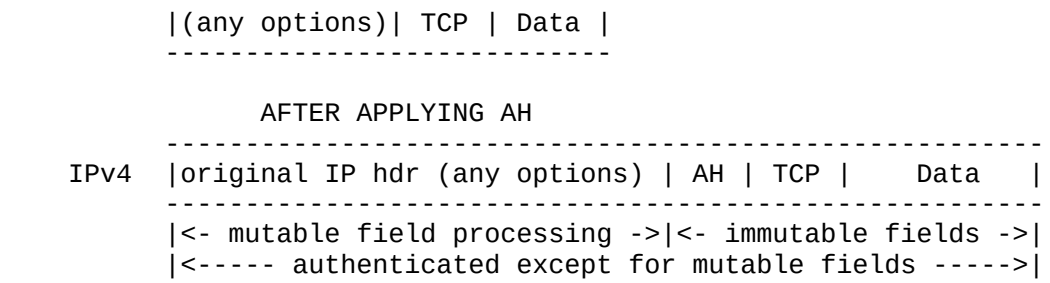
The Integrity Check Value is calculated over the contents of the packet and provides a means to check that all the protected bits were transferred successfully (i.e. without modification or unintentional corruption).

One of the major strengths of AH, which ESP does not also support, is that it protects the source IP address of the send. However, as a consequence it cannot be used when Network Address Translation is used since the source address will be modified, and AH is therefore of limited practical use.

Transport Mode

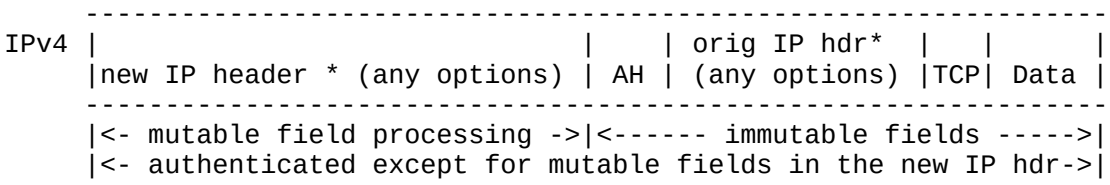
In transport mode, AH is inserted after the IP header and before a next layer protocol (e.g., TCP, UDP, ICMP, etc.) or before any other IPsec headers that have already been inserted. In the context of IPv4, this calls for placing AH after the IP header (and any options that it contains), but before the next layer protocol.





Tunnel Mode

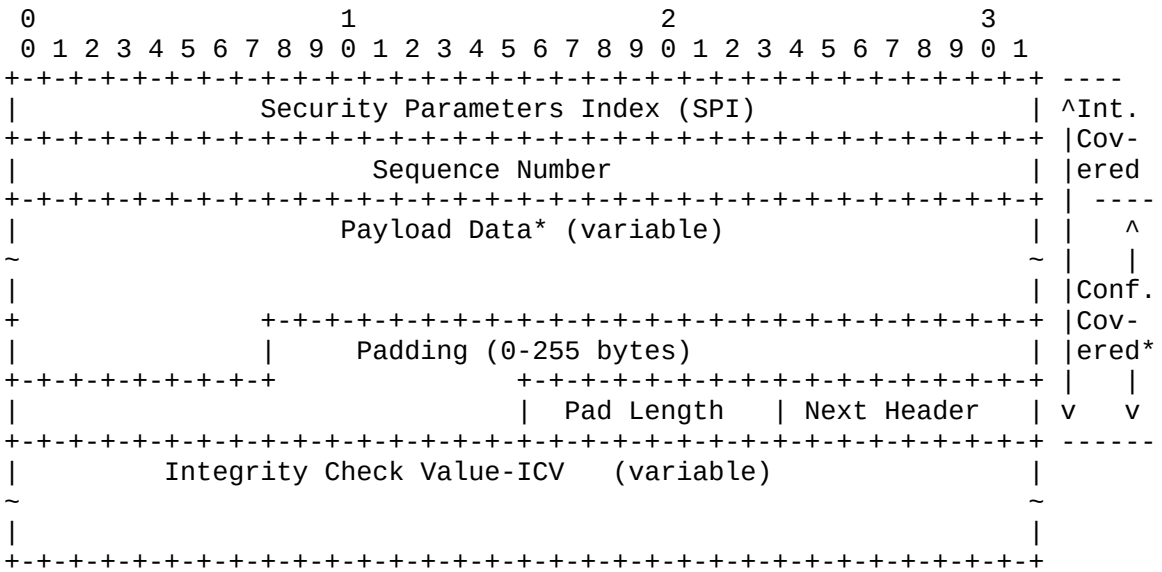
In tunnel mode, the "inner" IP header carries the ultimate (IP) source and destination addresses, while an "outer" IP header contains the addresses of the IPsec "peers," i.e., the addresses of security gateways.



ESP

Provides authentication, integrity and confidentiality, protocol 50, replay protection

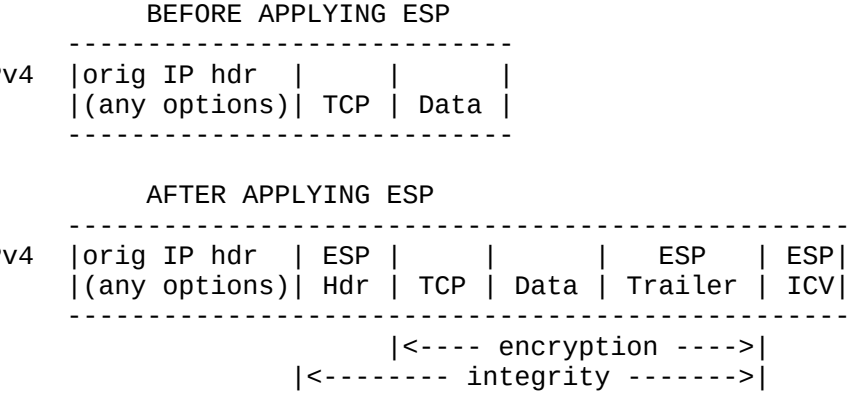
ESP headers satisfy all of the required characteristics for a VPN that were described above, although the authentication aspect is not as comprehensive as that provided by AH. ESP traffic is specified in RFC4303 and the format of the packet is shown in the figure below:



Note the introduction of a variable length padding length which provides observers with another obstacle to overcome when attempting to decode the applied encryption.

Transport Mode Processing

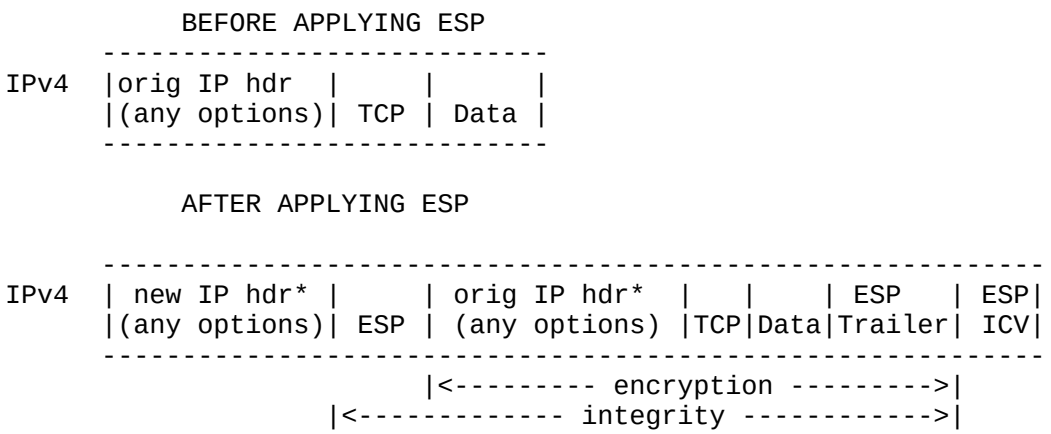
In transport mode, ESP is inserted after the IP header and before a next layer protocol, e.g., TCP, UDP, ICMP, etc. In the context of IPv4, this translates to placing ESP after the IP header (and any options that it contains), but before the next layer protocol.



Tunnel Mode Processing

In tunnel mode, the "inner" IP header carries the ultimate (IP) source and destination addresses, while an "outer" IP header contains the addresses of the IPsec "peers", e.g., addresses of security gateways.

Note that this means that the firewall on a receiving gateway will encounter such packets twice – once with the outer IP header and a second time with the original IP header following decryption of the original packet. Two sets of filter rules will be required to adequately filter such traffic.



Practical Examples

Practical configurations/deployments and considerations?

General Setup

Kernel with IPsec support

Install IPsec-utils (either from source or pre-compiled)

Choose the applicable example

Change configuration files as required

Change Firewall and forwarding on modem (both for site2site)

Change Firewall on the VPN gateway server

Start racoon daemons

Attempt to establish link

Debug

Verify secure traffic once established

Issues to be considered when deploying a VPN include:

1. the choice of encryption algorithms – DES is known to be weak and should not be used
2. collisions have been demonstrated for the non-HMAC variants of both MD5 and SHA1 hashing algorithms, although SHA1 is believed to offer better protection.
3. AH is not compatible with NAT
4. NAT requires further UDP encapsulation.
5. Modems require their firewall opening to allow UDP encapsulated IPsec traffic to pass. Usually UDP ports 500 and 4500 need to be opened and the traffic forwarded onto the Linux server that will terminate the IPsec traffic.
6. Internet Service Providers and/or 3rd parties (hotels) do not always pass IPsec traffic within their networks.
7. IPsec encapsulation adds some overhead to the original IP packet. Tunnel mode adds a minimum of 20 bytes and the MTU of the VPN gateway interface needs to be set

appropriately to ensure that fragmentation of packets does not occur (fragmentation is generally not well supported throughout the internet).

8. RFC that describes recommended encryption/hashing algos (RFC4308 – “Cryptographic Suites for IPsec”)

9. Use of ESP to provide confidentiality without integrity is NOT RECOMMENDED.

10.

Ensuring that you have a secure VPN

Unique site-to-site pre-shared keys

Strong pre-shared keys (length, character mixes)

IKE/ISAKMP aggressive mode – use main unless unsupported (eg iPhone with PSK)

Encryption with authentication

Use PFS (Perfect Forward Secrecy)

Strong encryption/authentication algorithms (tradeoff is MIPS)

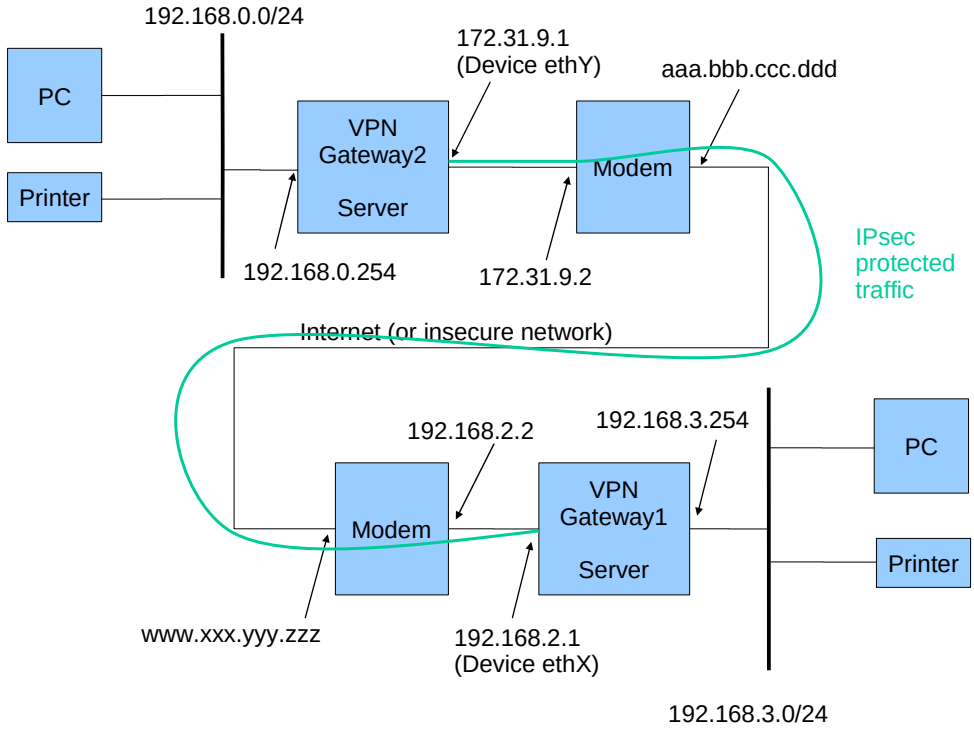
- Encryption - at least 3DES, preferably AES
- Authentication - at least MD5-HMAC, preferably SHA1-HMAC

Don't use DH group 1

Necessary configuration files

- Config file
- Script

Site-to-site Scenario



Use cases:

- Offsite-backup
- Multiple clients at each endpoint
- Multi-way office communications

Considerations:

- NAT-T – reasons for choosing config XYZ to allow NAT-T (e.g. not AH etc!), nat keepalive
- Need statically assigned external IPs for modems
- Need modems to forward allowed traffic to their respective gateway

NAS, Printer, DNLA

Rsync Backup

Racoon configuration files:

The files found in the Appendix will need transferring to the appropriate machines and modifying using a series of search/replace commands to change the IP addresses to match the systems on which they are to be deployed.

Firewall

Both the external modem and the gateway computers firewall need to be opened to allow traffic to pass on the following UDP ports

- 500 - IKE
- 4500 - NAT-T

Both modems also need to be configured to forward UDP traffic on ports 500 and 4500 on to their respective VPN gateway (i.e. Modem attached to Gateway1 will forward to address 192.168.2.1)

Protocols 50,51

Two-passes for packets

```
# Gateway 1
# VPN server details
INTERNETIFACE=ethX
VPN_REMOTE_SERVER=aaa.bbb.ccc.ddd/32
VPN_REMOTE_SUBNET=192.168.0.0/24
VPN_REMOTE_LOCAL=172.31.9.1/32
VPN_LOCAL_SUBNET=192.168.3.0/24
# Allow External UDP encapsulated packets
$IPTABLES -A INPUT -i $INTERNETIFACE -p udp --dport 500 -j ACCEPT
$IPTABLES -A INPUT -i $INTERNETIFACE -p udp --dport 4500 -j ACCEPT
# Allow Internal decrypted packets
$IPTABLES -A INPUT -i $INTERNETIFACE -s $VPN_REMOTE_SERVER -j ACCEPT
$IPTABLES -A INPUT -i $INTERNETIFACE -s $VPN_REMOTE_SUBNET -j ACCEPT
$IPTABLES -A INPUT -i $INTERNETIFACE -s $VPN_REMOTE_LOCAL -j ACCEPT
$IPTABLES -A FORWARD -i $INTERNETIFACE -s $VPN_REMOTE_LOCAL -d $VPN_LOCAL_SUBNET -j ACCEPT


# Gateway 2
# VPN server details
INTERNETIFACE=ethY
VPN_REMOTE_SERVER=www.xxx.yyy.zzz/32
VPN_REMOTE_SUBNET=192.168.3.0/24
VPN_REMOTE_LOCAL=192.168.2.1/32
VPN_LOCAL_SUBNET=192.168.0.0/24
# Allow External UDP encapsulated packets
$IPTABLES -A INPUT -i $INTERNETIFACE -p udp --dport 500 -j ACCEPT
$IPTABLES -A INPUT -i $INTERNETIFACE -p udp --dport 4500 -j ACCEPT
# Allow Internal decrypted packets
$IPTABLES -A INPUT -i $INTERNETIFACE -s $VPN_REMOTE_SERVER -j ACCEPT
$IPTABLES -A INPUT -i $INTERNETIFACE -s $VPN_REMOTE_SUBNET -j ACCEPT
$IPTABLES -A INPUT -i $INTERNETIFACE -s $VPN_REMOTE_LOCAL -j ACCEPT
$IPTABLES -A FORWARD -i $INTERNETIFACE -s $VPN_REMOTE_LOCAL -d $VPN_LOCAL_SUBNET -j ACCEPT
```

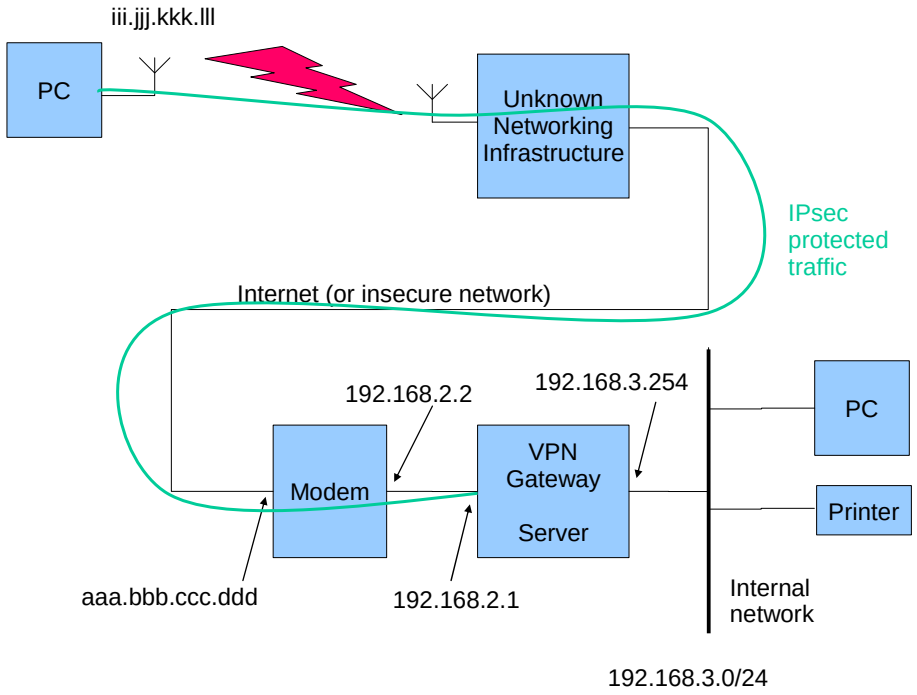
Routing

What to discuss here?

Creating CA and Certificates

What to put where (CAs, etc.); remove passphrase from keys

Road-warrior Scenario



Use cases:

Connecting home from a hotel or to the office from home

Racoon configuration files:

The files found in the Appendix will need transferring to the gateway machine and modifying using a series of search/replace commands to change the IP addresses to match the systems on which they are to be deployed.

PSK.txt

Just DON'T use this method!

Firewall – rp_filter

Firewall

modem and gateway

UDP ports

- 500 - IKE
- 4500 - NAT-T

Protocols 50,51

Two-passes for packets

```
# VPN server details
INTERNETIFACE=ethx
VPN_ROADWARRIOR=192.168.9.248/30
SERVERIP=192.168.2.1
FWD_DEST=0.0.0.0/0
# External UDP encapsulated packets
$IPTABLES -A INPUT -i $INTERNETIFACE -p udp --dport 500 -j ACCEPT
$IPTABLES -A INPUT -i $INTERNETIFACE -p udp --dport 4500 -j ACCEPT
# Internal decrypted packets
$IPTABLES -A INPUT -i $INTERNETIFACE -s $VPN_ROADWARRIOR -d $SERVERIP -j ACCEPT
$IPTABLES -A FORWARD -i $INTERNETIFACE -s $VPN_ROADWARRIOR -d $FWD_DEST -j ACCEPT
```

Routing

Temporary IP address assigned to the roadwarrior – how to get packets back to this device if it's not covered by the default route.

Racoonctl

Anything to mention?

Split-routing on client (use of home printer, NAS)

Creating CA and Certificates

VPN client software Shrew.net – see appendix for an example client configuration

Appendix

ipsec-tools

Building and Installing from source

Sourceforge

0.7.3 ipsec-tools

```
$ zypper in kernel-sources
$ zypper in krb5-devel
$ zypper in pam_radius
```

building it yourself:

```
$ ./configure --disable-shared --sysconfdir=/etc/racoon \
       --sharedstatedir=/var/run --localstatedir=/var/run \
       --enable-dpd --enable-hybrid --enable-frag \
       --enable-natt=yes --enable-gssapi=yes --enable-stats=yes \
       --enable-adminport --with-libpam --enable-security-context=no\
       --enable-fastquit
$ make
$ make check
$ make install
```

Or:

zypper in ipsec-tools

Certificate Generation

```
$ cd /etc/racoon
$ mkdir certs
$ cd certs

#################################################################
# Making the CA
#################################################################
$ /usr/share/ssl/misc/CA.sh -newca

CN = vpngateway.example.com


$ cd demoCA/
# Extend the lifetime of your CA to 10 years
$ openssl x509 -in cacert.pem -days 3650 -out cacert.pem -signkey ./private/cakey.pem
$ cd ..

#################################################################
# Making the certificate for the gateway
#################################################################
$ /usr/share/ssl/misc/CA.sh -newreq
$ /usr/share/ssl/misc/CA.sh -sign

mv newcert.pem vpngate_cert.pem
mv newkey.pem vpngate_key.pem
mv newreq.pem vpngate_req.pem

# Creating a windows compatible DER file for import [optional]
$ openssl x509 -in vpngate_cert.pem -outform DER -out vpngate_cert.der

# Make a key with no passphrase
$ openssl rsa -in vpngate_key.pem -out vpngate_key_np.pem

#################################################################
# Copying the CA into the openssl trusted certificates area
#################################################################
$ cd /etc/ssl/certs
$ cp /etc/racoon/certs/demoCA/cacert.pem MyCA.pem
/etc/racoon/certs/certlink.sh MyCA.pem

#################################################################
# Checking the certificate is correctly verified
#################################################################

$openssl verify vpngate_cert.pem
```

```
#################################################################
# Setup for a bidirectional gateway
#################################################################
$ cp Other_CA.pem /etc/ssl/certs
$ cd /etc/ssl/certs
$ certlink.sh Other_CA.pem

# Check that the gateway certificate I'll send you is now
# correctly verified by your machine:

$ openssl verify Other_vpngate_cert.pem
```

After the server credentials have been created, you will need to move the server certificate and private key files to the certificate path specified in your racoon configuration file.

The certificate authority public certificate should be given to each user that will be connecting to the gateway.

http://www.ipsec-howto.org/x595.html

/usr/share/ssl/misc/CA.sh

Making a CRL:

$ cd demoCA

$ echo “00” > crlnumber

$ cd ..

$ openssl ca -gencrl -out crl.pem

VPN Debug

Logging – racoon and firewall

Tcpdump / wireshark

Wireshark (hub vs switch; mirrored port)

Racoon log dissection

Begin with PSK

Begin with short lifetimes – 7 min and 19 minutes

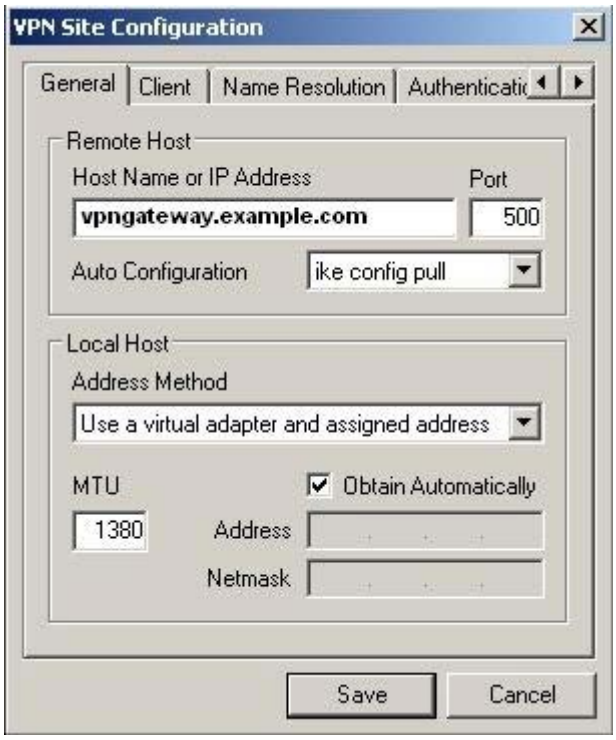
http://www.icsalabs.com/white_papers

Reference IPsec-utils Howto

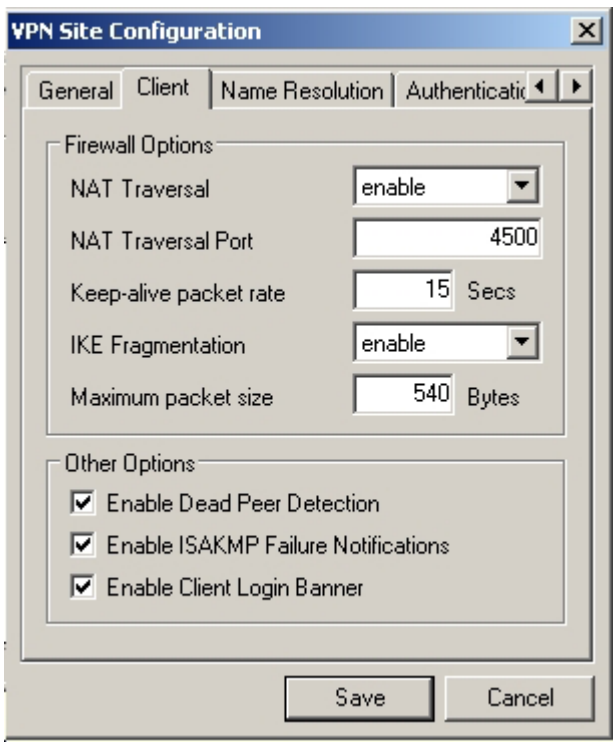
Example Windows XP Client configuration using Shrewnet Client

General settings tab.

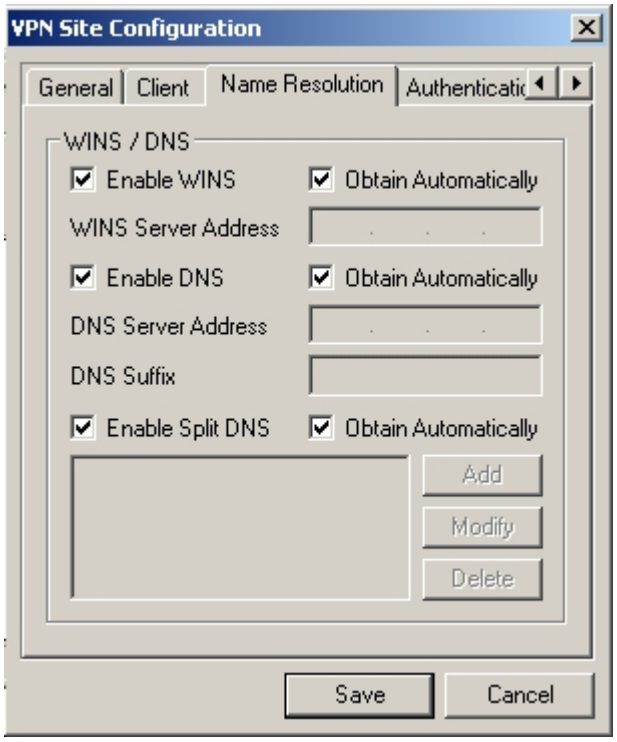
Enter the VPN gateways' name or IP address. Also note the addressing method.



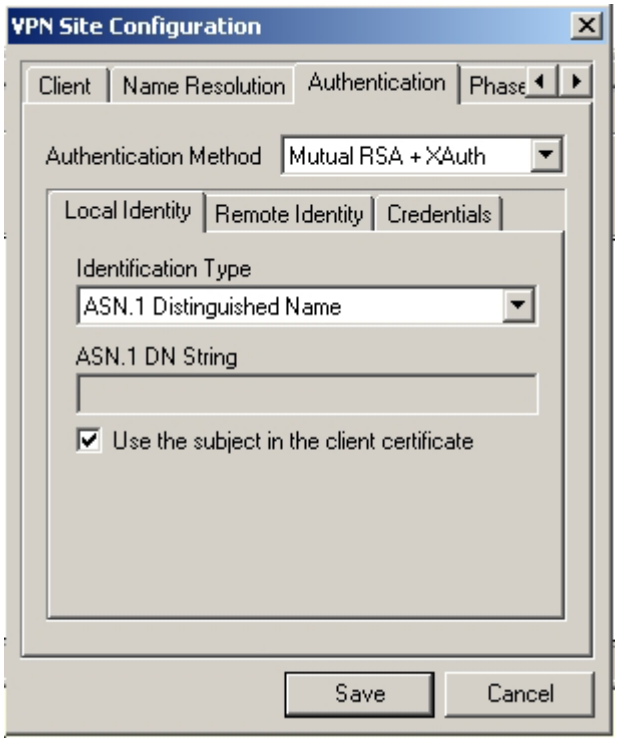
Client tab:



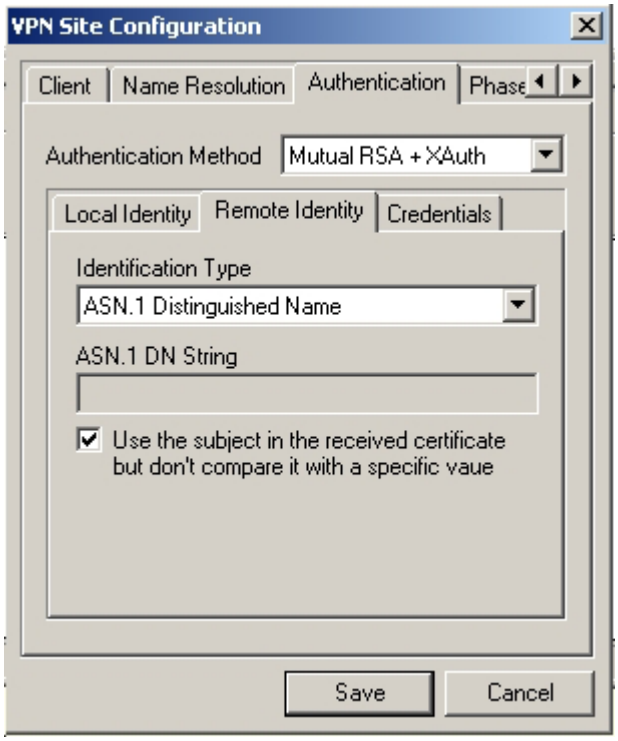
Name Resolution Tab:



Authentication tab, local identity sub-tab:

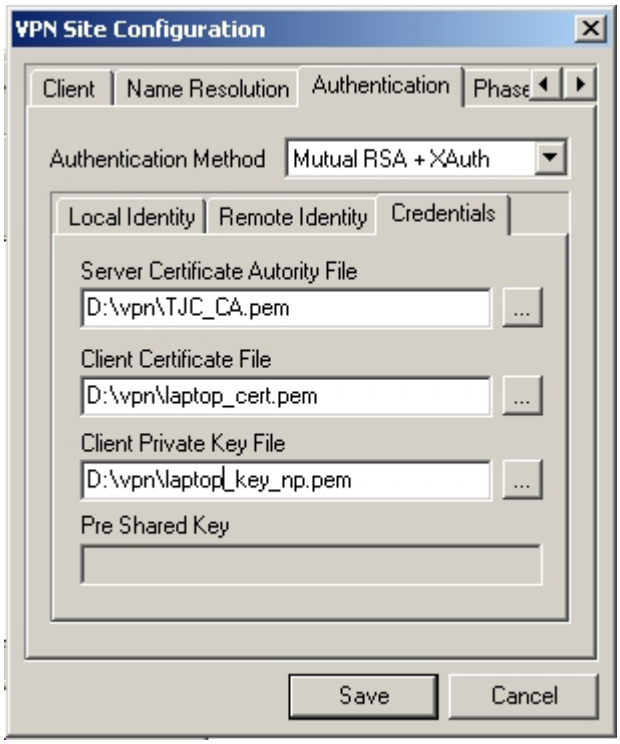


Authentication tab, remote identity sub-tab:

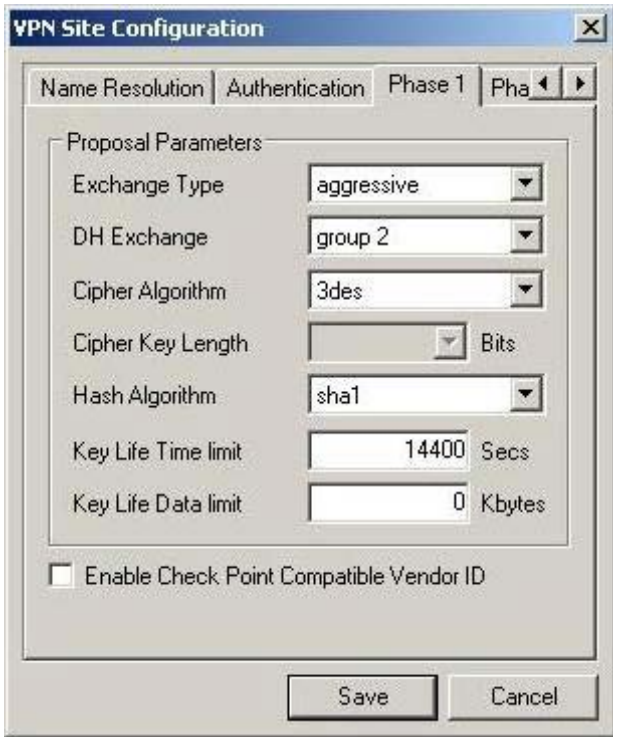


Authentication tab, credentials sub-tab:

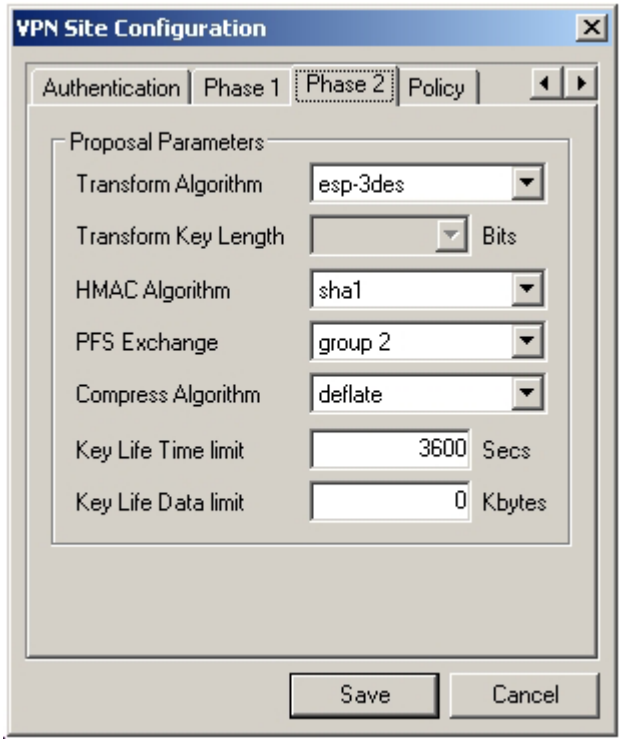
It is important that the paths to the certificate and key files are short and don't contain spaces!



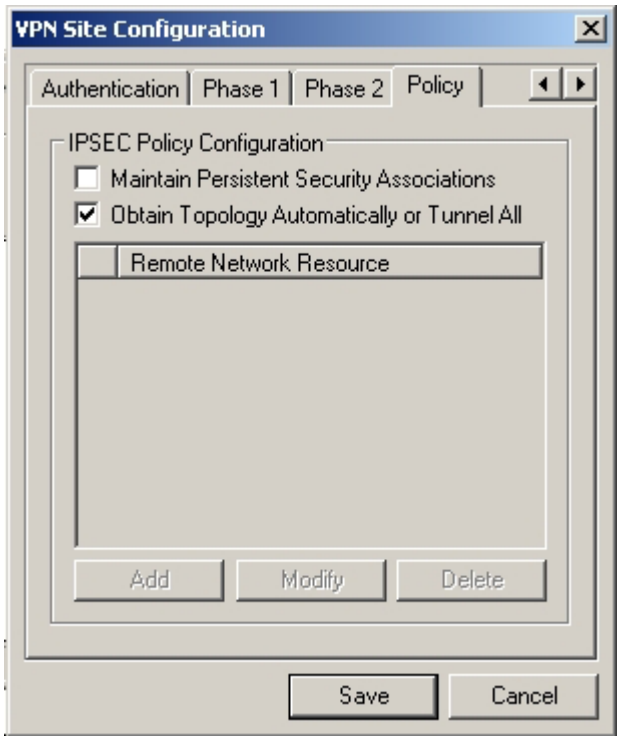
Phase1 tab:



Phase2 tab:



Policy tab:



Configuration files – Site2site, gateway1, racoon.conf

```
# ------------------------------------------------------------------------------
# RACOON configuration
# ------------------------------------------------------------------------------

# ------------------------------------------------------------------------------
# PATH and FILE specifications
# ------------------------------------------------------------------------------
# This section specifies various paths used by racoon.  When running in
# privilege separation mode, certificate and script paths are mandatory. A
# racoon restart is required if you want path changes to be taken into
# account. "path" must be defined before it can be used.
path include "/etc/racoon";

# search this file for pre_shared_key with various ID key.
path pre_shared_key "/etc/racoon/psk.txt";

# racoon will look for certificate file in the directory specified
# if a certificate/certificate request payload is received.
path certificate "/etc/racoon/certs";

# specifies a file in which SA information which is negotiated by racoon can be stored.
# racoon will install SA(s) from the file if the command-line option -B is specified.
# The file will increase in size because racoon simply adds SAs to the file.
# You should maintain the file manually.
##path backupsa "/etc/racoon/backupsa";

# racoon will search this directory for scripts hooks.  If you
# run with privilege separation, racoon(8) will refuse to execute a
# script stored outside of this directory.
##path script "/etc/racoon/scripts";

# Specifies file where to store PID of process.  If path starts
# with / it is treated as an absolute path. Otherwise, it is
# treated as a relative path to the VARRUN directory specified at
# compilation time.  Default is racoon.pid.
##path pidfile "/var/run/racoon.pid" ;

# ------------------------------------------------------------------------------
# LOGGING
# ------------------------------------------------------------------------------

# "log" specifies logging level.  It is followed by either "info", "notify", "debug"
# or "debug2".
log info;


# ------------------------------------------------------------------------------
# PRIVILEGE SEPARATION
# ------------------------------------------------------------------------------

# Specifies privilege separation parameters.  When enabled, these
# enable racoon to operate with an unprivileged instance doing
# most of the work, while a privileged instance takes care of per
# forming the following operations as root: reading PSK and private
# keys, launching hook scripts, and validating passwords against
# system databases or against PAM.  Please note that using privilege
# separation makes changes to the listen and paths sections
# ignored upon configuration reloads.  A racoon restart is
# required if you want such changes to be taken into account.
##privsep
##{
      # The user to which the unprivileged instance of racoon,
      # should switch.  This can be a quoted user name or a
      # numeric UID.
      ##user joebloggs;

      # The group the unprivileged instance of racoon, should
      # switch.  This can be a quoted group name or a numeric
      # GID.
      ##group users;

      # A directory to which the unprivileged instance of
      # racoon should chroot.  This directory should hold a
      # tree where the following files must be reachable:
      #         /dev/random
      # /dev/urandom
      # The certificates
      # The file containing the Xauth banner
      # The PSK file, the private keys, and the hook scripts are
      # accessed through the privileged instance of racoon and
      # do not need to be reachable in the chrooted tree.
      ##chroot "/var/lib/racoond";

##}

# ------------------------------------------------------------------------------
```

```
# PADDING
# ------------------------------------------------------------------------------

# This section defines the parameters related to padding.
padding
{
    # Defines a maximum padding length.  If randomize_length is
    # off, this is ignored.  The default is 20 bytes.
    maximum_length 20;

    # Enables the use of a randomized value for padding.  The
    # default is on.
    randomize off;

    # The pad length will be random.  The default is off.
    randomize_length off;

    # A means to constrain the peer to set the number of pad
    # bytes.  The default is off.
    strict_check off;

    # A means to put the number of pad bytes minus one into the
    # last part of the padding.  The default is on.
    exclusive_tail on;
}

# ------------------------------------------------------------------------------
# LISTENING
# ------------------------------------------------------------------------------

# if no listen directive is specified, racoon will listen to all
# available interface addresses. When running in privilege separation mode,
# you need to restart racoon to have changes to the listen section taken into
# account.
listen
{
    # ISAKMP listening address and optional port; 500 is the IANA allocated port
    isakmp 192.168.2.1 [500];

    # ISAKMP NAT-Traversal address and ports - 4500 is the IANA allocated port
    isakmp_natt 192.168.2.1 [4500];

    # The path, owner, and group values specify the socket path, owner, and group.
    # They must be quoted. The defaults are /var/racoon/racoon.sock, UID 0, and GID 0.
    # mode is the access mode in octal. The default is 0600.
    ##adminsock disabled ;
    adminsock "/var/run/racoon/racoon.sock" "root" "root" 600;

    # Require that all addresses for ISAKMP must be bound.
    # This statement will be ignored if you do not specify any addresses.
    strict_address;

}

# ------------------------------------------------------------------------------
# TIMERS
# ------------------------------------------------------------------------------

# Specification of the default values for various timers.
# These value can be changed per remote node.
timer
{
    # The interval between sending NAT-Traversal keep-alive
    # packets.  The default time is 20 seconds.  Set to 0s to
    # disable keep-alive packets.
    natt_keepalive  20 sec;

    # The maximum number of retries to send.  The default is 5.
    counter 5;

    # The interval to resend, in seconds.  The default time is
    # 10 seconds.
    interval 10 sec;

    # The number of packets per send.  The default is 1.
    persend 1;

    # The maximum time it should take to complete phase 1.  The
    # default time is 15 seconds.
    phase1 30 sec;

    # The maximum time it should take to complete phase 2.  The
    # default time is 10 seconds.
    phase2 15 sec;

}

# ------------------------------------------------------------------------------
# Specific Road-warrior and Site configurations
```

```
# --------------------------------------------------------------------------

# Include any site or road-warrior specific configurations here
include "site_gw1.conf" ;
```

Configuration files – Site2site, gateway1, site_gw1.conf

```
# ------------------------------------------------------------------------------
# Site-to-Site configuration for site gw1 (www.xxx.yyy.zzz)
# ------------------------------------------------------------------------------


 remote aaa.bbb.ccc.ddd
 {
       # Defines the exchange mode for phase 1 when racoon is the
       # initiator.  It also means the acceptable exchange mode
       # when racoon is the responder.  More than one mode can be
       # specified by separating them with a comma.  All of the
       # modes are acceptable.  The first exchange mode is what
       # racoon uses when it is the initiator.
       # Do NOT use AGRESSIVE mode unless you have to as it is less secure.
       # options : main, aggressive, base
       exchange_mode main;

       # Forces the use of IPsec DOI (Domain of Interpretation) as specified RFC 2407. You may omit this
       # statement. Useful when creating a VPN with different endpoints, ie from different manufacturers/types.
       doi ipsec_doi;

       # Forces use of SIT_IDENTITY_ONLY as specified RFC 2407. You may omit this statement.
       # All implementations supporting DOIs must support SIT_IDENTITY_ONLY.
       situation identity_only;

       # This directive enables use of the NAT-Traversal IPsec extension (NAT-T).
       # NAT-T allows one or both peers to reside behind a NAT gateway (i.e., doing address-
       # or port-translation).  If a NAT gateway is detected during the phase 1 handshake,
       # racoon will attempt to negotiate the use of NAT-T with the remote peer.  If the negotiation
       # succeeds, all ESP and AH packets for the given connection will be encapsulated into UDP
       # datagrams (port 4500, by default).  Possible values are:
       # on      NAT-T is used when a NAT gateway is detected
       #         between the peers.
       # off     NAT-T is not proposed/accepted.  This is the
       #         default.
       # force   NAT-T is used regardless of whether a NAT gateway
       #         is detected between the peers or not.
       # Please note that NAT-T support is a compile-time option. Although it is enabled in the
       # source distribution by default, it may not be available in your particular build.
       # In that case you will get a warning when using any NAT-T related config options.
       nat_traversal force ;

       # If you do not want to initiate the negotiation, set this to on.  The default value is off.
       # It is useful for a server.
       passive off ;

       # Enable this to send an INITIAL-CONTACT message. The default value is on.
       # This message is useful only when the responder implementation chooses an old SA when there
       # are multiple SAs with different established time and the initiator reboots.  If racoon did not
       # send the message, the responder would use an old SA even when a new SA was established.
       # For systems that use a KAME derived IPSEC stack, the sysctl(8) variable net.key.preferred_oldsa can
       # be used to control this preference.  When the value is zero, the stack always uses a new SA.
       initial_contact on ;

       # This directive is for the responder.  Therefore you should also set passive to on in order that
       # racoon only becomes a responder.  If the responder does not have any policy in SPD during
       # phase 2 negotiation, and the directive is set to on, then racoon will choose the first proposal
       # in the SA payload from the initiator, and generate policy entries from the proposal.  It is
       # useful to negotiate with clients whose IP address is allocated dynamically.  Note that an
       # inappropriate policy might be installed into the responder's SPD by the initiator, so other
       # communications might fail if such policies are installed due to a policy mismatch between the initiator
       # and the responder.
       # on and require values mean the same thing (generate a require policy).
       # unique tells racoon to set up unique policies, with a monotonically increasing reqid number
       # (between 1 and IPSEC_MANUAL_REQID_MAX). This should be the case when multiple roadwarriors may attach.
       # This directive is ignored in the initiator case.
       # The default value is off. Others are on, require and unique.
       generate_policy off ;

       # ------------------------------------------------------------------------------
       # CERTIFICATES and AUTHENTICATION RELATED
       # ------------------------------------------------------------------------------

       # Determine the type and location of certificates to be used.
       ##certificate_type plain_rsa "privkeyfile" ;
       ##certificate_type x509 "certificate filename" "private key filename";
       certificate_type x509 "vpngate_cert.pem" "vpngate_key_np.pem" ;

       # CA
       # file name of the root certificate authority.  Default is /etc/openssl/cert.pem
       ca_type x509 "SDP_CA.pem" ;

       # By default, the identifier sent by the remote host (as specified in its
       # my_identifier statement) is compared with the credentials in the
       # certificate used to authenticate the remote host as follows:
       # Type asn1dn: The entire certificate subject name is compared
```

```
# with the identifier, e.g. "C=XX, O=YY, ...".
# Type address, fqdn, or user_fqdn: The certificate's subjectAltName
# is compared with the identifier. If the two do not match the negotiation
# will fail. If you do not want to verify the identifier using the peer's
# certificate, set this to off. Generally we can't set this on because
# racoon attempts to verify with a root CA which most of us don't have, so ...
# was verify_cert off;
verify_cert on;

# Specifies the identifier sent to the remote host and the
# type to use in the phase 1 negotiation.  address, fqdn,
# user_fqdn, keyid, and asn1dn can be used as an idtype.
# The qualifier is currently only used for keyid, and can
# be either file or tag.
# Options are:
# address ["www.xxx.yyy.zzz"]; - IP address; this is the default
# user_fqdn "tim@fqdn.com"; - User Fully Qualified Domain Name
# fqdn "vpnserver.fqdn.com"; - Fully Qualified Domain Name
# keyid "file"; - KEY_ID
# asn1dn "string"; ASN.1 distinguished name from the certificate in use.
# Defaults to the DN from the certificates Subject field if string is omitted
my_identifier asn1dn ;

# Specifies the peer's identifier to be received.  If it is not defined then
# racoon will not verify the peer's identifier in ID payload transmitted
# from the peer.  If it is defined, the behavior of the verification depends
# on the flag of verify_identifier.  The usage of idtype is the same as
# my_identifier except that the individual component values of an asn1dn
# identifier may specified as * to match any value (e.g. "C=XX, O=MyOrg, OU=*, CN=Mine").
# Alternative acceptable peer identifiers may be specified
# by repeating the peers_identifier statement.
## peers_identifier asn1dn "C=UK,ST=*,L=*,O=*,CN=*,emailAddress=*" ;
peers_identifier asn1dn ;

# If you want to verify the peer's identifier, set this to
# on. In this case, if the value defined by
# peers_identifier is not the same as the peer's identifier
# in the ID payload, the negotiation will fail.  The
# default is off.
verify_identifier on;

# Specifies the login to use in client-side Hybrid authentication.  It is available
# only if racoon has been built with this option.  The associated password is
# looked up in the pre-shared key files, using the login string as the key id.
## xauth_login [string];

# If dnssec is defined, racoon will ignore the CERT payload from the peer,
# and try to get the peer's certificate from DNS instead. If certfile is defined,
# racoon will ignore the CERT payload from the peer, and will use this
# certificate as the peer's certificate.  If plain_rsa is defined, racoon
# will expect pubkeyfile to be the peer's public key that was generated by plainrsa-gen.
##peers_certfile (dnssec | certfile | plain_rsa pubkeyfile);

# If you do not want to send a certificate, set this to off.  The default is on.
##send_cert off;

# If you do not want to send a certificate request, set this to off.  The default is on.
##send_cr off;

# ------------------------------------------------------------------------------
# SCRIPT RELATED
# ------------------------------------------------------------------------------

# Gather network information through ISAKMP mode configuration.  Default is off.
# This is useful if scripts are called which require additional information in which to operate correctly.
##mode_cfg on ;

# Shell scripts that get executed when a phase 1 SA goes up or down.
# Both scripts get either phase1_up or phase1_down as first argument, and the following variables
# are set in their environment:
# LOCAL_ADDR The local address of the phase 1 SA.
# LOCAL_PORT The local port used for IKE for the phase 1 SA.
# REMOTE_ADDR The remote address of the phase 1 SA.
# REMOTE_PORT The remote port used for IKE for the phase 1 SA.
# The following variables are only set if mode_cfg was enabled:
# INTERNAL_ADDR4 An IPv4 internal address obtained by ISAKMP mode config.
# INTERNAL_NETMASK4 An IPv4 internal netmask obtained by ISAKMP mode config.
# INTERNAL_CIDR4 An IPv4 internal netmask obtained by ISAKMP mode config, in CIDR notation.
# INTERNAL_DNS4 The first internal DNS server IPv4 address obtained by ISAKMP mode config.
# INTERNAL_DNS4_LIST A list of internal DNS servers IPv4 address obtained by ISAKMP mode config,
# separated by spaces.
# INTERNAL_WINS4 The first internal WINS server IPv4 address obtained by ISAKMP mode config.
# INTERNAL_WINS4_LIST A list of internal WINS servers IPv4 address obtained by ISAKMP mode config,
# separated by spaces.
# SPLIT_INCLUDE The space separated list of IPv4 addresses and masks (address slash mask) that define the
# networks to be encrypted (as opposed to the default where all the traffic should be encrypted) ;
# obtained by ISAKMP mode config ; SPLIT_INCLUDE and SPLIT_LOCAL are mutually exclusive.
# SPLIT_LOCAL The space separated list of IPv4 addresses and masks (address slash mask) that define the
# networks to be considered local, and thus excluded from the tunnels ; obtained by ISAKMP mode config.
```

```
# DEFAULT_DOMAIN The DNS default domain name obtained by ISAKMP mode config.
##script script phase1_up
##script script phase1_down

# ----------------------------------------------------------------------------
# DEAD PEER DETECTION
# ----------------------------------------------------------------------------

# Dead peer detection timeout. This option activates the DPD and sets the time (in seconds)
# allowed between 2 proof of liveliness requests. The default value is 0, which disables
# DPD monitoring, but still negotiates DPD support.
dpd_delay 30;

# If dpd_delay is set, this sets the delay (in seconds) to wait for a proof of liveliness
# before considering it as failed and send another request.  The default value is 5.
dpd_retry 5;

# If dpd_delay is set, this sets the maximum number of liveliness proofs to request
# (without reply) before considering the peer is dead.  The default value is 5.
dpd_maxfail 3;

# define the byte size of nonce value.  Racoon can send any value although RFC2409 specifies
# that the value MUST be between 8 and 256 bytes.  The default size is 16 bytes.
nonce_size 16;

# An optional number to identify the remote proposal and to link it only with sainfos
# which have the same number. Defaults to 0.
##ph1id number;

# ----------------------------------------------------------------------------
# FRAGMENTATION
# ----------------------------------------------------------------------------

# Enable receiver-side IKE fragmentation if racoon has been built with this feature.
# If set to on, racoon will advertise itself as being capable of receiving packets split by IKE
# fragmentation.
# This extension is there to work around broken firewalls that do not work with fragmented UDP packets.
# IKE fragmentation is always enabled on the sender-side, and it is used if the peer advertises itself as
# IKE fragmentation capable.  By selecting force, IKE Fragmentation will be used when racoon is acting as
# the initiator even before the remote peer has advertised itself as IKE fragmentation capable.
ike_frag on ;

# This option is only relevant if you use NAT traversal in tunnel mode.  Its purpose is to work around
# broken DSL routers that reject UDP fragments, by fragmenting the IP packets before ESP encapsulation.
# The result is ESP over UDP of fragmented packets instead of fragmented ESP over UDP packets
# (i.e., IP:UDP:ESP:frag(IP) instead of frag(IP:UDP:ESP:IP)). fraglen is the maximum size of the fragments.
# 552 should work anywhere, but the larger fraglen is, the better the performance.
# Note that because PMTU discovery is broken on many sites, you will have to use MSS clamping if you want
# TCP to work correctly.
##esp_frag 552;

# ----------------------------------------------------------------------------
# PROPOSAL RELATED
# ----------------------------------------------------------------------------

# Ensure that the lifetime and PFS (Perfect Forwarding Secrecy - relies on Diffie Helmann group match at
# both ends
# Specifies the action of lifetime length, key length and PFS of the phase 2 selection
# on the responder side, and the action of lifetime check in phase 1.  The default level is strict.
# If the level is:
#        obey    The responder will obey the initiator anytime.
#        strict  If the responder's lifetime length is longer than the initiator's or the responder's
#                key length is shorter than the initiator's, the responder will use the initiator's value.
#                Otherwise, the proposal will be rejected.  If PFS is not required by the responder,
#                the responder will obey the proposal. If PFS is required by both sides and the responder's
#                group is not equal to the initiator's, then the responder will reject the proposal.
#        claim   If the responder's lifetime length is longer than the initiator's or the responder's key
#                length is shorter than the initiator's, the responder will use the initiator's value.
#                If the responder's lifetime length is shorter than the initiator's, the responder uses its
#                own length AND sends a RESPONDER-LIFETIME notify message to an initiator in the case of
#                lifetime (phase 2 only).  For PFS, this directive behaves the same as strict.
#        exact   If the initiator's lifetime or key length is not equal to the responder's, the responder
#                will reject the proposal. If PFS is required by both sides and the responder's group is
#                not equal to the initiator's, then the responder will reject the proposal.
proposal_check strict ;

# Tells racoon to act on unencrypted deletion messages during phase 1.
# This is a small security risk, so the default is off, meaning that racoon will keep on trying
# to establish a connection even if the user credentials are wrong, for instance.
##weak_phase1_check off;

# If this value is set to on, then both values of ID payloads in the phase 2 exchange are always used as
# the addresses of end-point of IPsec-SAs.  The default is off.
##support_proxy off ;

# An optional number to identify the remote proposal and to link it only with sainfos who have the same
# number. Defaults to 0.
##ph1id 0;
```

```
        # Set up IKE policy
        proposal {
                # Specifies the encryption algorithm used for the phase 1 negotiation.  This directive must be
                # defined.  algorithm is one of following: des, 3des, blowfish, cast128, aes, camellia for Oakley.
                # For other transforms, this statement should not be used.
                encryption_algorithm aes;

                # Defines the hash algorithm used for the phase 1 negotiation.  This directive must be defined.
                # algorithm is one of following: md5, sha1, sha256, sha384, sha512 for Oakley.
                hash_algorithm sha1;

                # Defines the authentication method used for the phase 1 negotiation.  This directive must be
                # defined.  type is one of: pre_shared_key, rsasig for plain RSA authentication), gssapi_krb,
                # hybrid_rsa_server, hybrid_rsa_client, xauth_rsa_server, xauth_rsa_client, xauth_psk_server
                # or xauth_psk_client.
                # Certificate based authentication has more potential areas for mis-configuration.
                # Consequently pre_shared_keys are useful as a "simpler" debug approach.
                authentication_method rsasig ;

                # Defines the group used for the Diffie-Hellman exponentiations.  This directive must be defined.
                # group is one of following: modp768, modp1024, modp1536, modp2048, modp3072, modp4096, modp6144,
                # modp8192.  Alternatively you can define 1, 2, 5, 14, 15, 16, 17, or 18 as the DH group number.
                # If using aggressive mode, you must define the same DH group in each proposal.
                dh_group modp2048;

                # Defines the lifetime of the phase 1 SA proposal. Any proposal will be accepted, and the attribute
                # will not be proposed to the peer if you do not specify it (them).  They can be individually
                # specified in each proposal.
                lifetime time 28800 sec ;

                # Defines the GSS-API endpoint name, to be included as an attribute in the SA, if the gssapi_krb
                # authentication method is used.  If this is not defined, the default value of 'host/hostname' is
                # used, where hostname is the value returned by the hostname command.
                ## gss_id "vpnserver" ;

        }
 }

# ------------------------------------------------------------------------------
# IKE PHASE 2
# ------------------------------------------------------------------------------

# defines the parameters of the IKE phase 2 (IPsec-SA establishment).
# source_id and destination_id are constructed like:
# address address [/ prefix] [[port]] ul_proto
# or
# subnet address [/ prefix] [[port]] ul_proto
# or
# idtype string
#
# An id string should be expressed to match the exact value of an ID payload (source is the local end,
# destination is the remote end).  This is not like a filter rule.
# For example, if you define 3ffe:501:4819::/48 as source_id. 3ffe:501:4819:1000:/64 will not match.
# In the case of a longest prefix (selecting a single host), address instructs to send ID type of ADDRESS while
# subnet instructs to send ID type of SUBNET.  Otherwise, these instructions are identical.
# The group keyword allows an XAuth group membership check to be performed for this sainfo section.  When the
# mode_cfg auth source is set to system or ldap, the XAuth user is verified to be a member of the specified group
# before allowing a matching SA to be negotiated.

## local pool (192.168.3.0/24) and remote pool (192.168.0.0/24)
##sainfo address 192.168.3.0/24 any address 192.168.0.0/24 any
##{
        # define the group of Diffie-Hellman exponentiations.  If you do not require PFS then you can omit this
        # directive.
        # Any proposal will be accepted if you do not specify one. group is one of following: modp768, modp1024,
        # modp1536, modp2048, modp3072, modp4096, modp6144, modp8192.  Alternatively you can define 1, 2, 5, 14,
        # 15, 16, 17, or 18 as the DH group number.
        ##pfs_group modp768;

        # define how long an IPsec-SA will be used, in timeunits. Any proposal will be accepted, and no
        # attribute(s) will be proposed to the peer if you do not specify it(them). See the proposal_check
        # directive.
        ##lifetime time 3600 sec ;

        # racoon  does not have a list of security protocols to be negotiated.  The list of security protocols are
        # passed by SPD in the kernel.  Therefore you have to define all of the potential algorithms in the phase 2
        # proposals even if there are algorithms which will not be used.  These algorithms are define by using the
        # following three directives, with a single comma as the separator. For algorithms that can take
        # variable-length keys, algorithm names can be followed by a key length, like "blowfish 448".
        # racoon will compute the actual phase 2 proposals by computing the permutation of the specified
        # algorithms, and then combining them with the security protocol specified by the SPD.  For example, if
        # des, 3des, hmac_md5, and hmac_sha1 are specified as algorithms, we have four combinations for use with
        # ESP, and two for AH.  Then, based on the SPD settings, racoon will construct the actual proposals.  If
        # the SPD entry asks for ESP only, there will be 4 proposals. If it asks for both AH and ESP, there will
        # be 8 proposals.  Note that the kernel may not support the algorithm you have specified.
        # des, 3des, des_iv64, des_iv32, rc5, rc4, idea, 3idea, cast128, blowfish, null_enc, twofish, rijndael,
        # aes, camellia (used with ESP).
        ##encryption_algorithm aes;
```

```
        # des, 3des, des_iv64, des_iv32, hmac_md5, hmac_sha1, hmac_sha256, hmac_sha384, hmac_sha512,
        # non_auth (used with ESP authentication and AH)
        ##authentication_algorithm hmac_sha1;

        # deflate (used with IPComp)
        ##compression_algorithm deflate;
##}


## local pool (192.168.3.0/24) and remote pool (192.168.0.0/24)
sainfo address 192.168.3.0/24 any address 192.168.0.0/24 any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local pool (192.168.3.0/24) and remote server (aaa.bbb.ccc.ddd)
sainfo address 192.168.3.0/24 any address aaa.bbb.ccc.ddd any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local pool (192.168.3.0/24) and remote inet (172.31.9.1)
sainfo address 192.168.3.0/24 any address 172.31.9.1 any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local server (www.xxx.yyy.zzz) and remote pool (192.168.0.0/24)
sainfo address www.xxx.yyy.zzz any address 192.168.0.0/24 any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local server (www.xxx.yyy.zzz) and remote server (aaa.bbb.ccc.ddd)
sainfo address www.xxx.yyy.zzz any address aaa.bbb.ccc.ddd any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local server (www.xxx.yyy.zzz) and remote inet (172.31.9.1)
sainfo address www.xxx.yyy.zzz any address 172.31.9.1 any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

# local inet (192.168.2.1) and remote pool (192.168.0.0/24)
sainfo address 192.168.2.1 any address 192.168.0.0/24 any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local inet (192.168.2.1) and remote server (aaa.bbb.ccc.ddd)
sainfo address 192.168.2.1 any address aaa.bbb.ccc.ddd any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}
```

```
## local inet (192.168.2.1) and remote inet (172.31.9.1)
sainfo address 192.168.2.1 any address 172.31.9.1 any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}
```

Configuration files – Site2site, gateway1, setkey.conf

```
#!/usr/local/sbin/setkey -f
# First of all flush the SPD database
flush;
spdflush;
# Security policies
#
## remote pool (192.168.0.0/24) and local pool (192.168.3.0/24)
spdadd 192.168.0.0/24 192.168.3.0/24 any -P in ipsec esp/tunnel/aaa.bbb.ccc.ddd-192.168.2.1/require;
spdadd 192.168.3.0/24 192.168.0.0/24 any -P out ipsec esp/tunnel/192.168.2.1-aaa.bbb.ccc.ddd/require;

## remote server (aaa.bbb.ccc.ddd) and local pool (192.168.3.0/24)
spdadd aaa.bbb.ccc.ddd 192.168.3.0/24 any -P in ipsec esp/tunnel/aaa.bbb.ccc.ddd-192.168.2.1/require;
spdadd 192.168.3.0/24 aaa.bbb.ccc.ddd any -P out ipsec esp/tunnel/192.168.2.1-aaa.bbb.ccc.ddd/require;

## remote inet (172.31.9.1) and local pool (192.168.3.0/24)
spdadd 172.31.9.1 192.168.3.0/24 any -P in ipsec esp/tunnel/aaa.bbb.ccc.ddd-192.168.2.1/require;
spdadd 192.168.3.0/24 172.31.9.1 any -P out ipsec esp/tunnel/192.168.2.1-aaa.bbb.ccc.ddd/require;

## remote pool (192.168.0.0/24) and local inet (192.168.2.1)
spdadd 192.168.0.0/24 192.168.2.1 any -P in ipsec esp/tunnel/aaa.bbb.ccc.ddd-192.168.2.1/require;
spdadd 192.168.2.1 192.168.0.0/24 any -P out ipsec esp/tunnel/192.168.2.1-aaa.bbb.ccc.ddd/require;

## remote server (aaa.bbb.ccc.ddd) and local inet (192.168.2.1)
spdadd aaa.bbb.ccc.ddd 192.168.2.1 any -P in ipsec esp/tunnel/aaa.bbb.ccc.ddd-192.168.2.1/require;
spdadd 192.168.2.1 aaa.bbb.ccc.ddd any -P out ipsec esp/tunnel/192.168.2.1-aaa.bbb.ccc.ddd/require;

## remote inet (172.31.9.1) and local inet (192.168.2.1)
spdadd 172.31.9.1 192.168.2.1 any -P in ipsec esp/tunnel/aaa.bbb.ccc.ddd-192.168.2.1/require;
spdadd 192.168.2.1 172.31.9.1 any -P out ipsec esp/tunnel/192.168.2.1-aaa.bbb.ccc.ddd/require;
```

Configuration files – Site2site, gateway2, racoon.conf

```
# ------------------------------------------------------------------------------
# RACOON configuration
# ------------------------------------------------------------------------------

# ------------------------------------------------------------------------------
# PATH and FILE specifications
# ------------------------------------------------------------------------------
# This section specifies various paths used by racoon.  When running in
# privilege separation mode, certificate and script paths are mandatory. A
# racoon restart is required if you want path changes to be taken into
# account. "path" must be defined before it can be used.
path include "/etc/racoon";

# search this file for pre_shared_key with various ID key.
path pre_shared_key "/etc/racoon/psk.txt";

# racoon will look for certificate file in the directory specified
# if a certificate/certificate request payload is received.
path certificate "/etc/racoon/certs";

# specifies a file in which SA information which is negotiated by racoon can be stored.
# racoon will install SA(s) from the file if the command-line option -B is specified.
# The file will increase in size because racoon simply adds SAs to the file.
# You should maintain the file manually.
##path backupsa "/etc/racoon/backupsa";

# racoon will search this directory for scripts hooks.  If you
# run with privilege separation, racoon(8) will refuse to execute a
# script stored outside of this directory.
##path script "/etc/racoon/scripts";

# Specifies file where to store PID of process.  If path starts
# with / it is treated as an absolute path. Otherwise, it is
# treated as a relative path to the VARRUN directory specified at
# compilation time.  Default is racoon.pid.
##path pidfile "/var/run/racoon.pid" ;

# ------------------------------------------------------------------------------
# LOGGING
# ------------------------------------------------------------------------------

# "log" specifies logging level.  It is followed by either "info", "notify", "debug"
# or "debug2".
#log info;
log debug;


# ------------------------------------------------------------------------------
# PRIVILEGE SEPARATION
# ------------------------------------------------------------------------------

# Specifies privilege separation parameters.  When enabled, these
# enable racoon to operate with an unprivileged instance doing
# most of the work, while a privileged instance takes care of per
# forming the following operations as root: reading PSK and private
# keys, launching hook scripts, and validating passwords against
# system databases or against PAM.  Please note that using privilege
# separation makes changes to the listen and paths sections
# ignored upon configuration reloads.  A racoon restart is
# required if you want such changes to be taken into account.
##privsep
##{
      # The user to which the unprivileged instance of racoon,
      # should switch.  This can be a quoted user name or a
      # numeric UID.
      ##user joebloggs;

      # The group the unprivileged instance of racoon, should
      # switch.  This can be a quoted group name or a numeric
      # GID.
      ##group users;

      # A directory to which the unprivileged instance of
      # racoon should chroot.  This directory should hold a
      # tree where the following files must be reachable:
      #         /dev/random
      # /dev/urandom
      # The certificates
      # The file containing the Xauth banner
      # The PSK file, the private keys, and the hook scripts are
      # accessed through the privileged instance of racoon and
      # do not need to be reachable in the chrooted tree.
      ##chroot "/var/lib/racoond";

##}

# ------------------------------------------------------------------------------
```

```
# PADDING
# ------------------------------------------------------------------------------

# This section defines the parameters related to padding.
padding
{
    # Defines a maximum padding length.  If randomize_length is
    # off, this is ignored.  The default is 20 bytes.
    maximum_length 20;

    # Enables the use of a randomized value for padding.  The
    # default is on.
    randomize off;

    # The pad length will be random.  The default is off.
    randomize_length off;

    # A means to constrain the peer to set the number of pad
    # bytes.  The default is off.
    strict_check off;

    # A means to put the number of pad bytes minus one into the
    # last part of the padding.  The default is on.
    exclusive_tail on;
}

# ------------------------------------------------------------------------------
# LISTENING
# ------------------------------------------------------------------------------

# if no listen directive is specified, racoon will listen to all
# available interface addresses. When running in privilege separation mode,
# you need to restart racoon to have changes to the listen section taken into
# account.
listen
{
    # ISAKMP listening address and optional port; 500 is the IANA allocated port
    isakmp 172.31.9.1 [500];

    # ISAKMP NAT-Traversal address and ports - 4500 is the IANA allocated port
    isakmp_natt 172.31.9.1 [4500];

    # The path, owner, and group values specify the socket path, owner, and group.
    # They must be quoted. The defaults are /var/racoon/racoon.sock, UID 0, and GID 0.
    # mode is the access mode in octal. The default is 0600.
    ##adminsock disabled ;
    adminsock "/var/run/racoon/racoon.sock" "root" "root" 600;

    # Require that all addresses for ISAKMP must be bound.
    # This statement will be ignored if you do not specify any addresses.
    strict_address;

}

# ------------------------------------------------------------------------------
# TIMERS
# ------------------------------------------------------------------------------

# Specification of the default values for various timers.
# These value can be changed per remote node.
timer
{
    # The interval between sending NAT-Traversal keep-alive
    # packets.  The default time is 20 seconds.  Set to 0s to
    # disable keep-alive packets.
    natt_keepalive  20 sec;

    # The maximum number of retries to send.  The default is 5.
    counter 5;

    # The interval to resend, in seconds.  The default time is
    # 10 seconds.
    interval 10 sec;

    # The number of packets per send.  The default is 1.
    persend 1;

    # The maximum time it should take to complete phase 1.  The
    # default time is 15 seconds.
    phase1 30 sec;

    # The maximum time it should take to complete phase 2.  The
    # default time is 10 seconds.
    phase2 15 sec;

}

# ------------------------------------------------------------------------------
# Specific Road-warrior and Site configurations
```

```
# --------------------------------------------------------------------------

# Include any site or road-warrior specific configurations here
include "site_gw2.conf" ;
```

Configuration files – Site2site, gateway2, site_gw2.conf

```
# ------------------------------------------------------------------------------
# Site-to-Site configuration for gateway2 (aaa.bbb.ccc.ddd)
# ------------------------------------------------------------------------------


remote www.xxx.yyy.zzz
{
	# Defines the exchange mode for phase 1 when racoon is the
	# initiator.  It also means the acceptable exchange mode
	# when racoon is the responder.  More than one mode can be
	# specified by separating them with a comma.  All of the
	# modes are acceptable.  The first exchange mode is what
	# racoon uses when it is the initiator.
	# Do NOT use AGRESSIVE mode unless you have to as it is less secure.
	# options : main, aggressive, base
	exchange_mode main;

	# Forces the use of IPsec DOI (Domain of Interpretation) as specified RFC 2407. You may omit this
	# statement. Useful when creating a VPN with different endpoints, ie from different manufacturers/types.
	doi ipsec_doi;

	# Forces use of SIT_IDENTITY_ONLY as specified RFC 2407. You may omit this statement.
	# All implementations supporting DOIs must support SIT_IDENTITY_ONLY.
	situation identity_only;

	# This directive enables use of the NAT-Traversal IPsec extension (NAT-T).
	# NAT-T allows one or both peers to reside behind a NAT gateway (i.e., doing address-
	# or port-translation).  If a NAT gateway is detected during the phase 1 handshake,
	# racoon will attempt to negotiate the use of NAT-T with the remote peer.  If the negotiation
	# succeeds, all ESP and AH packets for the given connection will be encapsulated into UDP
	# datagrams (port 4500, by default).  Possible values are:
	# on      NAT-T is used when a NAT gateway is detected
	#         between the peers.
	# off     NAT-T is not proposed/accepted.  This is the
	#         default.
	# force   NAT-T is used regardless of whether a NAT gateway
	#         is detected between the peers or not.
	# Please note that NAT-T support is a compile-time option. Although it is enabled in the
	# source distribution by default, it may not be available in your particular build.
	# In that case you will get a warning when using any NAT-T related config options.
	nat_traversal on ;

	# If you do not want to initiate the negotiation, set this to on.  The default value is off.
	# It is useful for a server.
	passive off ;

	# Enable this to send an INITIAL-CONTACT message. The default value is on.
	# This message is useful only when the responder implementation chooses an old SA when there
	# are multiple SAs with different established time and the initiator reboots.  If racoon did not
	# send the message, the responder would use an old SA even when a new SA was established.
	# For systems that use a KAME derived IPSEC stack, the sysctl(8) variable net.key.preferred_oldsa can
	# be used to control this preference.  When the value is zero, the stack always uses a new SA.
	initial_contact on ;

	# This directive is for the responder.  Therefore you should also set passive to on in order that
	# racoon only becomes a responder.  If the responder does not have any policy in SPD during
	# phase 2 negotiation, and the directive is set to on, then racoon will choose the first proposal
	# in the SA payload from the initiator, and generate policy entries from the proposal.  It is
	# useful to negotiate with clients whose IP address is allocated dynamically.  Note that an
	# inappropriate policy might be installed into the responder's SPD by the initiator, so other
	# communications might fail if such policies are installed due to a policy mismatch between the initiator
	# and the responder.
	# on and require values mean the same thing (generate a require policy).
	# unique tells racoon to set up unique policies, with a monotonically increasing reqid number
	# (between 1 and IPSEC_MANUAL_REQID_MAX). This should be the case when multiple roadwarriors may attach.
	# This directive is ignored in the initiator case.
	# The default value is off. Others are on, require and unique.
	## was generate_policy off ;
	generate_policy off ;

	# ------------------------------------------------------------------------------
	# CERTIFICATES and AUTHENTICATION RELATED
	# ------------------------------------------------------------------------------

	# Determine the type and location of certificates to be used.
	##certificate_type plain_rsa "privkeyfile" ;
	##certificate_type x509 "certificate filename" "private key filename";
	certificate_type x509 "vpngate_cert.pem" "vpngate_key_np.pem" ;

	# CA
	# file name of the root certificate authority.  Default is /etc/openssl/cert.pem
	ca_type x509 "TJC_CA.pem" ;

	# By default, the identifier sent by the remote host (as specified in its
	# my_identifier statement) is compared with the credentials in the
	# certificate used to authenticate the remote host as follows:
	# Type asn1dn: The entire certificate subject name is compared
	# with the identifier, e.g. "C=XX, O=YY, ...".
```

```
# Type address, fqdn, or user_fqdn: The certificate's subjectAltName
# is compared with the identifier. If the two do not match the negotiation
# will fail. If you do not want to verify the identifier using the peer's
# certificate, set this to off. Generally we can't set this on because
# racoon attempts to verify with a root CA which most of us don't have, so ...
verify_cert on;

# Specifies the identifier sent to the remote host and the
# type to use in the phase 1 negotiation.  address, fqdn,
# user_fqdn, keyid, and asn1dn can be used as an idtype.
# The qualifier is currently only used for keyid, and can
# be either file or tag.
# Options are:
# address ["www.xxx.yyy.zzz"]; - IP address; this is the default
# user_fqdn "tim@fqdn.com"; - User Fully Qualified Domain Name
# fqdn "vpnserver.fqdn.com"; - Fully Qualified Domain Name
# keyid "file"; - KEY_ID
# asn1dn "string"; ASN.1 distinguished name from the certificate in use.
# Defaults to the DN from the certificates Subject field if string is omitted
my_identifier asn1dn ;

# Specifies the peer's identifier to be received.  If it is not defined then
# racoon will not verify the peer's identifier in ID payload transmitted
# from the peer.  If it is defined, the behavior of the verification depends
# on the flag of verify_identifier.  The usage of idtype is the same as
# my_identifier except that the individual component values of an asn1dn
# identifier may specified as * to match any value (e.g. "C=XX, O=MyOrg, OU=*, CN=Mine").
# Alternative acceptable peer identifiers may be specified
# by repeating the peers_identifier statement.
# peers_identifier asn1dn "C=UK, ST=*, L=*, O=*, OU=*, CN=*, emailAddress=*" ;
peers_identifier asn1dn ;

# If you want to verify the peer's identifier, set this to
# on. In this case, if the value defined by
# peers_identifier is not the same as the peer's identifier
# in the ID payload, the negotiation will fail.  The
# default is off.
verify_identifier on;

# Specifies the login to use in client-side Hybrid authentication.  It is available
# only if racoon has been built with this option.  The associated password is
# looked up in the pre-shared key files, using the login string as the key id.
## xauth_login [string];

# If dnssec is defined, racoon will ignore the CERT payload from the peer,
# and try to get the peer's certificate from DNS instead. If certfile is defined,
# racoon will ignore the CERT payload from the peer, and will use this
# certificate as the peer's certificate.  If plain_rsa is defined, racoon
# will expect pubkeyfile to be the peer's public key that was generated by plainrsa-gen.
##peers_certfile (dnssec | certfile | plain_rsa pubkeyfile);

 # If you do not want to send a certificate, set this to off.  The default is on.
##send_cert off;

# If you do not want to send a certificate request, set this to off.  The default is on.
##send_cr off;

# ------------------------------------------------------------------------------
# SCRIPT RELATED
# ------------------------------------------------------------------------------

# Gather network information through ISAKMP mode configuration.  Default is off.
# This is useful if scripts are called which require additional information in which to operate correctly.
##mode_cfg on ;

# Shell scripts that get executed when a phase 1 SA goes up or down.
# Both scripts get either phase1_up or phase1_down as first argument, and the following variables
# are set in their environment:
# LOCAL_ADDR The local address of the phase 1 SA.
# LOCAL_PORT The local port used for IKE for the phase 1 SA.
# REMOTE_ADDR The remote address of the phase 1 SA.
# REMOTE_PORT The remote port used for IKE for the phase 1 SA.
# The following variables are only set if mode_cfg was enabled:
# INTERNAL_ADDR4 An IPv4 internal address obtained by ISAKMP mode config.
# INTERNAL_NETMASK4 An IPv4 internal netmask obtained by ISAKMP mode config.
# INTERNAL_CIDR4 An IPv4 internal netmask obtained by ISAKMP mode config, in CIDR notation.
# INTERNAL_DNS4 The first internal DNS server IPv4 address obtained by ISAKMP mode config.
# INTERNAL_DNS4_LIST A list of internal DNS servers IPv4 address obtained by ISAKMP mode config, separated
# by spaces.
# INTERNAL_WINS4 The first internal WINS server IPv4 address obtained by ISAKMP mode config.
# INTERNAL_WINS4_LIST A list of internal WINS servers IPv4 address obtained by ISAKMP mode config,
# separated by spaces.
# SPLIT_INCLUDE The space separated list of IPv4 addresses and masks (address slash mask) that define the
# networks to be encrypted (as opposed to the default where all the traffic should be encrypted) ;
# obtained by ISAKMP mode config ; SPLIT_INCLUDE and SPLIT_LOCAL are mutually exclusive.
# SPLIT_LOCAL The space separated list of IPv4 addresses and masks (address slash mask) that define the
# networks to be considered local, and thus excluded from the tunnels ; obtained by ISAKMP mode config.
# DEFAULT_DOMAIN The DNS default domain name obtained by ISAKMP mode config.
##script script phase1_up
```

```
##script script phase1_down

# ----------------------------------------------------------------------------
# DEAD PEER DETECTION
# ----------------------------------------------------------------------------

# Dead peer detection timeout. This option activates the DPD and sets the time (in seconds)
# allowed between 2 proof of liveliness requests. The default value is 0, which disables
# DPD monitoring, but still negotiates DPD support.
dpd_delay 30;

# If dpd_delay is set, this sets the delay (in seconds) to wait for a proof of liveliness
# before considering it as failed and send another request.  The default value is 5.
dpd_retry 5;

# If dpd_delay is set, this sets the maximum number of liveliness proofs to request
# (without reply) before considering the peer is dead.  The default value is 5.
dpd_maxfail 3;

# define the byte size of nonce value.  Racoon can send any value although RFC2409 specifies
# that the value MUST be between 8 and 256 bytes.  The default size is 16 bytes.
nonce_size 16;

# An optional number to identify the remote proposal and to link it only with sainfos
# which have the same number. Defaults to 0.
##ph1id number;

# ----------------------------------------------------------------------------
# FRAGMENTATION
# ----------------------------------------------------------------------------

# Enable receiver-side IKE fragmentation if racoon has been built with this feature.
# If set to on, racoon will advertise itself as being capable of receiving packets split by IKE
# fragmentation.
# This extension is there to work around broken firewalls that do not work with fragmented UDP packets.
# IKE fragmentation is always enabled on the sender-side, and it is used if the peer advertises itself as
# IKE fragmentation capable.  By selecting force, IKE Fragmentation will be used when racoon is acting as
# the initiator even before the remote peer has advertised itself as IKE fragmentation capable.
ike_frag on ;

# This option is only relevant if you use NAT traversal in tunnel mode.  Its purpose is to work around
# broken DSL routers that reject UDP fragments, by fragmenting the IP packets before ESP encapsulation.
# The result is ESP over UDP of fragmented packets instead of fragmented ESP over UDP packets
# (i.e., IP:UDP:ESP:frag(IP) instead of frag(IP:UDP:ESP:IP)). fraglen is the maximum size of the fragments.
# 552 should work anywhere, but the larger fraglen is, the better the performance.
# Note that because PMTU discovery is broken on many sites, you will have to use MSS clamping if you want
# TCP to work correctly.
##esp_frag 552;

# ----------------------------------------------------------------------------
# PROPOSAL RELATED
# ----------------------------------------------------------------------------

# Ensure that the lifetime and PFS (Perfect Forwarding Secrecy - relies on Diffie Helmann group) match at
# both ends
# Specifies the action of lifetime length, key length and PFS of the phase 2 selection
# on the responder side, and the action of lifetime check in phase 1.  The default level is strict.
# If the level is:
#       obey    The responder will obey the initiator anytime.
#       strict  If the responder's lifetime length is longer than the initiator's or the responder's
#               key length is shorter than the initiator's, the responder will use the initiator's value.
#               Otherwise, the proposal will be rejected.  If PFS is not required by the responder,
#               the responder will obey the proposal. If PFS is required by both sides and the responder's
#               group is not equal to the initiator's, then the responder will reject the proposal.
#       claim   If the responder's lifetime length is longer than the initiator's or the responder's key
#               length is shorter than the initiator's, the responder will use the initiator's value.  If
#               the responder's lifetime length is shorter than the initiator's, the responder uses its
#               own length AND sends a RESPONDER-LIFETIME notify message to an initiator in the case of
#               lifetime (phase 2 only).  For PFS, this directive behaves the same as strict.
#       exact   If the initiator's lifetime or key length is not equal to the responder's, the responder
#               will reject the proposal.  If PFS is required by both sides and the responder's group is
#               not equal to the initiator's, then the responder will reject the proposal.
proposal_check strict ;

# Tells racoon to act on unencrypted deletion messages during phase 1.
# This is a small security risk, so the default is off, meaning that racoon will keep on trying
# to establish a connection even if the user credentials are wrong, for instance.
##weak_phase1_check off;

# If this value is set to on, then both values of ID payloads in the phase 2 exchange are always used as
# the addresses of end-point of IPsec-SAs.  The default is off.
##support_proxy off ;

# An optional number to identify the remote proposal and to link it only with sainfos who have the same
# number. Defaults to 0.
##ph1id 0;

# Set up IKE policy
proposal {
```

```
                # Specifies the encryption algorithm used for the phase 1 negotiation.  This directive must be
                # defined.  algorithm is one of following: des, 3des, blowfish, cast128, aes, camellia for Oakley.
                # For other transforms, this statement should not be used.
                encryption_algorithm aes;

                # Defines the hash algorithm used for the phase 1 negotiation.  This directive must be defined.
                # algorithm is one of following: md5, sha1, sha256, sha384, sha512 for Oakley.
                hash_algorithm sha1;

                # Defines the authentication method used for the phase 1 negotiation.  This directive must be
                # defined.  type is one of: pre_shared_key, rsasig for plain RSA authentication), gssapi_krb,
                # hybrid_rsa_server, hybrid_rsa_client, xauth_rsa_server, xauth_rsa_client, xauth_psk_server
                # or xauth_psk_client.
                # Certificate based authentication has more potential areas for mis-configuration.
                # Consequently pre_shared_keys are useful as a "simpler" debug approach.
                authentication_method rsasig ;

                # Defines the group used for the Diffie-Hellman exponentiations.  This directive must be defined.
                # group is one of following: modp768, modp1024, modp1536, modp2048, modp3072, modp4096, modp6144,
                # modp8192.  Alternatively you can define 1, 2, 5, 14, 15, 16, 17, or 18 as the DH group number.
                # If using aggressive mode, you must define the same DH group in each proposal.
                dh_group modp2048;

                # Defines the lifetime of the phase 1 SA proposal. Any proposal will be accepted, and the attribute
                # will not be proposed to the peer if you do not specify it (them).  They can be individually
                # specified in each proposal.
                lifetime time 28800 sec ;

                # Defines the GSS-API endpoint name, to be included as an attribute in the SA, if the gssapi_krb
                # authentication method is used.  If this is not defined, the default value of 'host/hostname' is
                # used, where hostname is the value returned by the hostname command.
                ## gss_id "vpnserver" ;

        }
 }

# ------------------------------------------------------------------------------
# IKE PHASE 2
# ------------------------------------------------------------------------------

# defines the parameters of the IKE phase 2 (IPsec-SA establishment).
# source_id and destination_id are constructed like:
# address address [/ prefix] [[port]] ul_proto
# or
# subnet address [/ prefix] [[port]] ul_proto
# or
# idtype string
#
# An id string should be expressed to match the exact value of an ID payload (source is the local end,
# destination is the remote end).  This is not like a filter rule.  For example, if you define
# 3ffe:501:4819::/48 as source_id. 3ffe:501:4819:1000:/64 will not match.
# In the case of a longest prefix (selecting a single host), address instructs to send ID type of ADDRESS while
# subnet instructs to send ID type of SUBNET.  Otherwise, these instructions are identical.
# The group keyword allows an XAuth group membership check to be performed for this sainfo section.  When the
# mode_cfg auth source is set to system or ldap, the XAuth user is verified to be a member of the specified group
# before allowing a matching SA to be negotiated.

## local pool (192.168.3.0/24) and remote pool (192.168.0.0/24)
##sainfo address 192.168.3.0/24 any address 192.168.0.0/24 any
##{
      # define the group of Diffie-Hellman exponentiations.  If you do not require PFS then you can omit this
      # directive. Any proposal will be accepted if you do not specify one. group is one of following: modp768,
      # modp1024, modp1536, modp2048, modp3072, modp4096, modp6144, modp8192.  Alternatively you can define 1, 2,
      # 5, 14, 15, 16, 17, or 18 as the DH group number.
      ##pfs_group modp768;

      # define how long an IPsec-SA will be used, in timeunits. Any proposal will be accepted, and no
      # attribute(s) will be proposed to the peer if you do not specify it(them). See the proposal_check
      # directive.
      ##lifetime time 3600 sec ;

      # racoon  does not have a list of security protocols to be negotiated.  The list of security protocols are
      # passed by SPD in the kernel.  Therefore you have to define all of the potential algorithms in the phase 2
      # proposals even if there are algorithms which will not be used.  These algorithms are define by using the
      # following three directives, with a single comma as the separator. For algorithms that can take variable-
      # length keys, algorithm names can be followed by a key length, like "blowfish 448". racoon will compute
      # the actual phase 2 proposals by computing the permutation of the specified algorithms, and then combining
      # them with the security protocol specified by the SPD.  For example, if des, 3des, hmac_md5, and hmac_sha1
      # are specified as algorithms, we have four combinations for use with ESP, and two for AH.  Then, based on
      # the SPD settings, racoon will construct the actual proposals.  If the SPD entry asks for ESP only, there
      # will be 4 proposals. If it asks for both AH and ESP, there will be 8 proposals.  Note that the kernel may
      # not support the algorithm you have specified. des, 3des, des_iv64, des_iv32, rc5, rc4, idea, 3idea,
      # cast128, blowfish, null_enc, twofish, rijndael, aes, camellia (used with ESP).
      ##encryption_algorithm aes;

      # des, 3des, des_iv64, des_iv32, hmac_md5, hmac_sha1, hmac_sha256, hmac_sha384, hmac_sha512,
      # non_auth (used with ESP authentication and AH)
      ##authentication_algorithm hmac_sha1;
```

```
        # deflate (used with IPComp)
        ##compression_algorithm deflate;
##}


## local pool (192.168.0.0/24) and remote pool (192.168.3.0/24)
sainfo address 192.168.0.0/24 any address 192.168.3.0/24 any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local pool (192.168.0.0/24) and remote server (www.xxx.yyy.zzz)
sainfo address 192.168.0.0/24 any address www.xxx.yyy.zzz any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local pool (192.168.0.0/24) and remote inet (192.168.2.1)
sainfo address 192.168.0.0/24 any address 192.168.2.1 any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local server (aaa.bbb.ccc.ddd) and remote pool (192.168.3.0/24)
sainfo address aaa.bbb.ccc.ddd any address 192.168.3.0/24 any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local server (aaa.bbb.ccc.ddd) and remote server (www.xxx.yyy.zzz)
sainfo address aaa.bbb.ccc.ddd any address www.xxx.yyy.zzz any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local server (aaa.bbb.ccc.ddd) and remote inet (192.168.2.1)
sainfo address aaa.bbb.ccc.ddd any address 192.168.2.1 any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local inet (172.31.9.1) and remote pool (192.168.3.0/24)
sainfo address 172.31.9.1 any address 192.168.3.0/24 any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local inet (172.31.9.1) and remote server (www.xxx.yyy.zzz)
sainfo address 172.31.9.1 any address www.xxx.yyy.zzz any
{
        pfs_group modp2048;
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}

## local inet (172.31.9.1) and remote inet (192.168.2.1)
sainfo address 172.31.9.1 any address 192.168.2.1 any
{
        pfs_group modp2048;
```

```
        lifetime time 19 minutes ;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}
```

Configuration files – Site2site, gateway2, setkey.conf

```
#!/usr/local/sbin/setkey -f
# First of all flush the SPD database
flush;
spdflush;
## remote pool (192.168.3.0/24) and local pool (192.168.0.0/24)
spdadd 192.168.3.0/24 192.168.0.0/24 any -P in ipsec esp/tunnel/www.xxx.yyy.zzz-172.31.9.1/require;
spdadd 192.168.0.0/24 192.168.3.0/24 any -P out ipsec esp/tunnel/172.31.9.1-www.xxx.yyy.zzz/require;

## remote server (www.xxx.yyy.zzz) and local pool (192.168.0.0/24)
spdadd www.xxx.yyy.zzz 192.168.0.0/24 any -P in ipsec esp/tunnel/www.xxx.yyy.zzz-172.31.9.1/require;
spdadd 192.168.0.0/24 www.xxx.yyy.zzz any -P out ipsec esp/tunnel/172.31.9.1-www.xxx.yyy.zzz/require;

## remote inet (192.168.2.1) and local pool (192.168.0.0/24)
spdadd 192.168.2.1 192.168.0.0/24 any -P in ipsec esp/tunnel/www.xxx.yyy.zzz-172.31.9.1/require;
spdadd 192.168.0.0/24 192.168.2.1 any -P out ipsec esp/tunnel/172.31.9.1-www.xxx.yyy.zzz/require;

## remote pool (192.168.3.0/24) and local inet (172.31.9.1)
spdadd 192.168.3.0/24 172.31.9.1 any -P in ipsec esp/tunnel/www.xxx.yyy.zzz-172.31.9.1/require;
spdadd 172.31.9.1 192.168.3.0/24 any -P out ipsec esp/tunnel/172.31.9.1-www.xxx.yyy.zzz/require;

## remote server (www.xxx.yyy.zzz) and local inet (172.31.9.1)
spdadd www.xxx.yyy.zzz 172.31.9.1 any -P in ipsec esp/tunnel/www.xxx.yyy.zzz-172.31.9.1/require;
spdadd 172.31.9.1 www.xxx.yyy.zzz any -P out ipsec esp/tunnel/172.31.9.1-www.xxx.yyy.zzz/require;

## remote inet (192.168.2.1) and local inet (172.31.9.1)
spdadd 192.168.2.1 172.31.9.1 any -P in ipsec esp/tunnel/www.xxx.yyy.zzz-172.31.9.1/require;
spdadd 172.31.9.1 192.168.2.1 any -P out ipsec esp/tunnel/172.31.9.1-www.xxx.yyy.zzz/require;
```

Configuration files – Roadwarrior, gateway1, racoon.conf

```
# ------------------------------------------------------------------------------
# RACOON configuration
# ------------------------------------------------------------------------------

# ------------------------------------------------------------------------------
# PATH and FILE specifications
# ------------------------------------------------------------------------------
# This section specifies various paths used by racoon.  When running in
# privilege separation mode, certificate and script paths are mandatory. A
# racoon restart is required if you want path changes to be taken into
# account. "path" must be defined before it can be used.
path include "/etc/racoon";

# search this file for pre_shared_key with various ID key.
path pre_shared_key "/etc/racoon/psk.txt";

# racoon will look for certificate file in the directory specified
# if a certificate/certificate request payload is received.
path certificate "/etc/racoon/certs";

# specifies a file in which SA information which is negotiated by racoon can be stored.
# racoon will install SA(s) from the file if the command-line option -B is specified.
# The file will increase in size because racoon simply adds SAs to the file.
# You should maintain the file manually.
##path backupsa "/etc/racoon/backupsa";

# racoon will search this directory for scripts hooks.  If you
# run with privilege separation, racoon(8) will refuse to execute a
# script stored outside of this directory.
##path script "/etc/racoon/scripts";

# Specifies file where to store PID of process.  If path starts
# with / it is treated as an absolute path. Otherwise, it is
# treated as a relative path to the VARRUN directory specified at
# compilation time.  Default is racoon.pid.
##path pidfile "/var/run/racoon.pid" ;

# ------------------------------------------------------------------------------
# LOGGING
# ------------------------------------------------------------------------------

# "log" specifies logging level.  It is followed by either "info", "notify", "debug"
# or "debug2".
log info;


# ------------------------------------------------------------------------------
# PRIVILEGE SEPARATION
# ------------------------------------------------------------------------------

# Specifies privilege separation parameters.  When enabled, these
# enable racoon to operate with an unprivileged instance doing
# most of the work, while a privileged instance takes care of per
# forming the following operations as root: reading PSK and private
# keys, launching hook scripts, and validating passwords against
# system databases or against PAM.  Please note that using privilege
# separation makes changes to the listen and paths sections
# ignored upon configuration reloads.  A racoon restart is
# required if you want such changes to be taken into account.
##privsep
##{
      # The user to which the unprivileged instance of racoon,
      # should switch.  This can be a quoted user name or a
      # numeric UID.
      ##user joebloggs;

      # The group the unprivileged instance of racoon, should
      # switch.  This can be a quoted group name or a numeric
      # GID.
      ##group users;

      # A directory to which the unprivileged instance of
      # racoon should chroot.  This directory should hold a
      # tree where the following files must be reachable:
      #         /dev/random
      # /dev/urandom
      # The certificates
      # The file containing the Xauth banner
      # The PSK file, the private keys, and the hook scripts are
      # accessed through the privileged instance of racoon and
      # do not need to be reachable in the chrooted tree.
      ##chroot "/var/lib/racoond";

##}

# ------------------------------------------------------------------------------
# PADDING
```

```
# ------------------------------------------------------------------------------

# This section defines the parameters related to padding.
padding
{
    # Defines a maximum padding length.  If randomize_length is
    # off, this is ignored.  The default is 20 bytes.
    maximum_length 20;

    # Enables the use of a randomized value for padding.  The
    # default is on.
    randomize off;

    # The pad length will be random.  The default is off.
    randomize_length off;

    # A means to constrain the peer to set the number of pad
    # bytes.  The default is off.
    strict_check off;

    # A means to put the number of pad bytes minus one into the
    # last part of the padding.  The default is on.
    exclusive_tail on;
}

# ------------------------------------------------------------------------------
# LISTENING
# ------------------------------------------------------------------------------

# if no listen directive is specified, racoon will listen to all
# available interface addresses. When running in privilege separation mode,
# you need to restart racoon to have changes to the listen section taken into
# account.
listen
{
    # ISAKMP listening address and optional port; 500 is the IANA allocated port
    isakmp 192.168.2.1 [500];

    # ISAKMP NAT-Traversal address and ports - 4500 is the IANA allocated port
    isakmp_natt 192.168.2.1 [4500];

    # The path, owner, and group values specify the socket path, owner, and group.
    # They must be quoted. The defaults are /var/racoon/racoon.sock, UID 0, and GID 0.
    # mode is the access mode in octal. The default is 0600.
    ##adminsock disabled ;
    adminsock "/var/run/racoon/racoon.sock" "root" "root" 600;

    # Require that all addresses for ISAKMP must be bound.
    # This statement will be ignored if you do not specify any addresses.
    strict_address;

}

# ------------------------------------------------------------------------------
# TIMERS
# ------------------------------------------------------------------------------

# Specification of the default values for various timers.
# These value can be changed per remote node.
timer
{
    # The interval between sending NAT-Traversal keep-alive
    # packets.  The default time is 20 seconds.  Set to 0s to
    # disable keep-alive packets.
    natt_keepalive  20 sec;

    # The maximum number of retries to send.  The default is 5.
    counter 5;

    # The interval to resend, in seconds.  The default time is
    # 10 seconds.
    interval 10 sec;

    # The number of packets per send.  The default is 1.
    persend 1;

    # The maximum time it should take to complete phase 1.  The
    # default time is 15 seconds.
    phase1 30 sec;

    # The maximum time it should take to complete phase 2.  The
    # default time is 10 seconds.
    phase2 15 sec;

}

# ------------------------------------------------------------------------------
# Specific Road-warrior and Site configurations
# ------------------------------------------------------------------------------
```

```
# Include any site or road-warrior specific configurations here
include "roadwarrior.conf" ;
```

Configuration files – Roadwarrior, gateway1, roadwarrior.conf

```
# ------------------------------------------------------------------------------
# ROADWARRIOR CONFIGURATION
# ------------------------------------------------------------------------------

remote anonymous
{

        # Defines the exchange mode for phase 1 when racoon is the
        # initiator.  It also means the acceptable exchange mode
        # when racoon is the responder.  More than one mode can be
        # specified by separating them with a comma.  All of the
        # modes are acceptable.  The first exchange mode is what
        # racoon uses when it is the initiator.
        # Do NOT use AGRESSIVE mode unless you have to as it is less secure.
        # options : main, aggressive, base
        exchange_mode aggressive;

        # If you do not want to initiate the negotiation, set this to on.  The default value is off.
        # It is useful for a server.
        passive on;

        # If you want to verify the peer's identifier, set this to
        # on. In this case, if the value defined by
        # peers_identifier is not the same as the peer's identifier
        # in the ID payload, the negotiation will fail.  The
        # default is off.
        verify_identifier on;

           # By default, the identifier sent by the remote host (as specified in its
        # my_identifier statement) is compared with the credentials in the
        # certificate used to authenticate the remote host as follows:
        # Type asn1dn: The entire certificate subject name is compared
        # with the identifier, e.g. "C=XX, O=YY, ...".
        # Type address, fqdn, or user_fqdn: The certificate's subjectAltName
        # is compared with the identifier. If the two do not match the negotiation
        # will fail. If you do not want to verify the identifier using the peer's
        # certificate, set this to off.
        verify_cert on;

        # Specifies the identifier sent to the remote host and the
        # type to use in the phase 1 negotiation.  address, fqdn,
        # user_fqdn, keyid, and asn1dn can be used as an idtype.
        # The qualifier is currently only used for keyid, and can
        # be either file or tag.
        # Options are:
        # address ["www.xxx.yyy.zzz"]; - IP address; this is the default
        # user_fqdn "tim@fqdn.com"; - User Fully Qualified Domain Name
        # fqdn "vpnserver.fqdn.com"; - Fully Qualified Domain Name
        # keyid "file"; - KEY_ID
        # asn1dn "string"; ASN.1 distinguished name from the certificate in use.
        # Defaults to the DN from the certificates Subject field if string is omitted
        my_identifier asn1dn ;

        # Specifies the peer's identifier to be received.  If it is not defined then
        # racoon will not verify the peer's identifier in ID payload transmitted
        # from the peer.  If it is defined, the behavior of the verification depends
        # on the flag of verify_identifier.  The usage of idtype is the same as
        # my_identifier except that the individual component values of an asn1dn
        # identifier may specified as * to match any value (e.g. "C=XX, O=MyOrg, OU=*, CN=Mine").
        # Alternative acceptable peer identifiers may be specified
        # by repeating the peers_identifier statement.
        ##peers_identifier user_fqdn "tim@home.com";
        peers_identifier asn1dn ;

        # Determine the type and location of certificates to be used.
        ##certificate_type plain_rsa "privkeyfile" ;
        ##certificate_type x509 "certificate filename" "private key filename";
        certificate_type x509 "vpngate_cert.pem" "vpngate_key_np.pem" ;

        # CA
        # file name of the root certificate authority.  Default is /etc/openssl/cert.pem
        ca_type x509 "demoCA/cacert.pem" ;

        # This directive is for the responder.  Therefore you should also set passive to on in order that
        # racoon only becomes a responder.  If the responder does not have any policy in SPD during
        # phase 2 negotiation, and the directive is set to on, then racoon will choose the first proposal
        # in the SA payload from the initiator, and generate policy entries from the proposal.  It is
        # useful to negotiate with clients whose IP address is allocated dynamically.  Note that an
        # inappropriate policy might be installed into the responder's SPD by the initiator, so other
        # communications might fail if such policies are installed due to a policy mismatch between the initiator
        # and the responder.
        # on, require values mean the same thing (generate a require policy).
        # unique tells racoon to set up unique policies, with a monotonically increasing reqid number
        # (between 1 and IPSEC_MANUAL_REQID_MAX). This should be the case when multiple roadwarriors may attach.
        # This directive is ignored in the initiator case.
        # The default value is off.
        generate_policy unique;
```

```
# Enable receiver-side IKE fragmentation if racoon has been built with this feature.
# If set to on, racoon will advertise itself as being capable of receiving packets split by IKE
# fragmentation.
# This extension is there to work around broken firewalls that do not work with fragmented UDP packets.
# IKE fragmentation is always enabled on the sender-side, and it is used if the peer advertises itself as
# IKE fragmentation capable.  By selecting force, IKE Fragmentation will be used when racoon is acting as
# the initiator even before the remote peer has advertised itself as IKE fragmentation capable.
ike_frag on;

# Gather network information through ISAKMP mode configuration.  Default is off.
# This is useful if scripts are called which require additional information in which to operate correctly.
mode_cfg on ;

# This directive enables use of the NAT-Traversal IPsec extension (NAT-T).
# NAT-T allows one or both peers to reside behind a NAT gateway (i.e., doing address-
# or port-translation).  If a NAT gateway is detected during the phase 1 handshake,
# racoon will attempt to negotiate the use of NAT-T with the remote peer.  If the negotiation
# succeeds, all ESP and AH packets for the given connection will be encapsulated into UDP
# datagrams (port 4500, by default).  Possible values are:
# on      NAT-T is used when a NAT gateway is detected
#         between the peers.
# off     NAT-T is not proposed/accepted.  This is the
#         default.
# force   NAT-T is used regardless of whether a NAT gateway
#         is detected between the peers or not.
# Please note that NAT-T support is a compile-time option. Although it is enabled in the
# source distribution by default, it may not be available in your particular build.
# In that case you will get a warning when using any NAT-T related configuration options.
nat_traversal on;

# Dead peer detection timeout. This option activates the DPD and sets the time (in seconds)
# allowed between 2 proof of liveliness requests. The default value is 0, which disables
# DPD monitoring, but still negotiates DPD support.
dpd_delay 30;

# If dpd_delay is set, this sets the delay (in seconds) to wait for a proof of liveliness
# before considering it as failed and send another request.  The default value is 5.
dpd_retry 10;

# If dpd_delay is set, this sets the maximum number of liveliness proofs to request
# (without reply) before considering the peer is dead.  The default value is 5.
dpd_maxfail 5;

# Defines the lifetime of the phase 1 SA proposal. Any proposal will be accepted, and the attribute
# will not be proposed to the peer if you do not specify it (them).  They can be individually
# specified in each proposal.
lifetime time 14400 sec ;

# Ensure that the lifetime and PFS (Perfect Forwarding Secrecy - relies on Diffie Helmann group) matching
# at both ends.
# Specifies the action of lifetime length, key length and PFS of the phase 2 selection
# on the responder side, and the action of lifetime check in phase 1.  The default level is strict.
# If the level is:
#        obey    The responder will obey the initiator anytime.
#        strict  If the responder's lifetime length is longer than the initiator's or the responder's
#                key length is shorter than the initiator's, the responder will use the initiator's value.
#                Otherwise, the proposal will be rejected.  If PFS is not required by the responder,
#                the responder will obey the proposal. If PFS is required by both sides and the responder's
#                group is not equal to the initiator's, then the responder will reject the proposal.
#        claim   If the responder's lifetime length is longer than the initiator's or the responder's key
#                length is shorter than the initiator's, the responder will use the initiator's value.  If
#                the responder's lifetime length is shorter than the initiator's, the responder uses its
#                own length AND sends a RESPONDER-LIFETIME notify message to an initiator in the case of
#                lifetime (phase 2 only).  For PFS, this directive behaves the same as strict.
#        exact   If the initiator's lifetime or key length is not equal to the responder's, the responder
#                will reject the proposal.  If PFS is required by both sides and the responder's group is
#                not equal to the initiator's, then the responder will reject the proposal.
proposal_check claim ;

# Set up IKE policy
proposal {

        # Specifies the encryption algorithm used for the phase 1 negotiation.  This directive must be
        # defined.  algorithm is one of following: des, 3des, blowfish, cast128, aes, camellia for Oakley.
        # For other transforms, this statement should not be used.
        encryption_algorithm 3des;

        # Defines the hash algorithm used for the phase 1 negotiation.  This directive must be defined.
        # algorithm is one of following: md5, sha1, sha256, sha384, sha512 for Oakley.
        hash_algorithm sha1;

        # Defines the authentication method used for the phase 1 negotiation.  This directive must be
        # defined.  type is one of: pre_shared_key, rsasig for plain RSA authentication), gssapi_krb,
        # hybrid_rsa_server, hybrid_rsa_client, xauth_rsa_server, xauth_rsa_client, xauth_psk_server
        # or xauth_psk_client.
        # Certificate based authentication has more potential areas for mis-configuration.
        # Consequently pre_shared_keys are useful as a "simpler" debug approach.
        authentication_method xauth_rsa_server;
```

```
                # Defines the group used for the Diffie-Hellman exponentiations.  This directive must be defined.
                # group is one of following: modp768, modp1024, modp1536, modp2048, modp3072, modp4096, modp6144,
                # modp8192.  Alternatively you can define 1, 2, 5, 14, 15, 16, 17, or 18 as the DH group number.
                # If using aggressive mode, you must define the same DH group in each proposal.
                dh_group modp1024;

        }
}

# ------------------------------------------------------------------------------
# ISAKMP mode configuration settings
# ------------------------------------------------------------------------------
mode_cfg
{
        # Specifies the source for IP addresses and netmask allocated through ISAKMP mode config.
        # local means to use the local IP pool defined by the network4 and pool_size statements.
        # This is the default.
        # radius means to use a RADIUS server.  It works only if racoon was built with libradius
        # support and requires RADIUS authentiation. RADIUS configuration is handled by radius.conf.
        # ldap means to use an LDAP server.  It works only if racoon was built with libldap support
        # and requires LDAP authentication.  LDAP configuration is handled by statements in
        # the ldapcfg section.
        conf_source local;

        # Enables or disables accounting for Xauth logins and logouts.  The default is none which disables
        # accounting.
        # Specifying system enables system accounting through utmp.
        # Specifying radius enables RADIUS accounting. It works only if racoon was built with libradius support
        # and requires RADIUS authentication.  RADIUS configuration is handled by radius.conf.
        # Specifying pam enables PAM accounting. It works only if racoon was built with libpam support and
        # requires PAM authentication.
        accounting none ;


        # The local IP pool base address from which dynamically allocated IPv4 addresses should be
        # taken.  This is used if conf_source is set to local or if the RADIUS server returned
        # 255.255.255.254.  Default is 0.0.0.0
        network4 192.168.9.248;

        # Specify the size of the IP address pool, either local or allocated through RADIUS.
        # conf_source selects the local pool or the RADIUS configuration, but in both configurations,
        # you cannot have more than size users connected at the same time.  The default is 255.
        pool_size 4;

        # The local IP pool network mask. This is used if conf_source is set to local or if the
        # RADIUS server returned 255.255.255.254.  Default is /0.0.0.0.
        netmask4 255.255.255.0;

        # A list of IPv4 addresses for WINS servers, separated by commas, or on multiple wins4 lines.
         wins4 192.168.7.1;

        # A list of IPv4 addresses for DNS servers, separated by commas, or on multiple dns4 lines.
        dns4 192.168.7.1;

        # The default DNS domain to send.
        default_domain "home.example.com";

        # The network configuration to send, in cidr notation (e.g. 192.168.1.0/24).
        # If include is specified, the tunnel should be only used to encrypt the cidr specified destinations;
        # otherwise, if local_lan is specified, everything will pass through the tunnel but the cidr specified
        # destinations.
        #split_network local_lan 192.168.7.1/24;

        # Specifies the source for authentication of users through Xauth.
        # system means to use the Unix user database.  This is the default.
        # radius means to use a RADIUS server.  It works only if racoon was built with libradius support.
        # Radius configuration is hanlded by radius.conf.
        # pam means to use PAM.  It works only if racoon was built with libpam support.
        # ldap means to use LDAP.  It works only if racoon was built with libldap support.
        # LDAP configuration is handled by statements in the ldapcfg section.
        auth_source system;

        # Specifies the group memberships for Xauth in quoted group name strings.
        # When defined, the authenticating user must be a member of at least one group for Xauth to succeed.
        auth_groups "vpn-users";

        # Specifies the source for group validataion of users through Xauth.
        # system means to use the Unix user database.  This is the default.
        # ldap means to use LDAP. It works only if racoon was built with libldap support and requires LDAP
        # authentication. LDAP configuration is handled by statements in the ldapcfg section.
        group_source system;

        # The path of a file displayed on the client at connection time.  Default is /etc/motd.
        banner "/etc/racoon/motd.txt";

        # On each failed Xauth authentication attempt, refuse new attempts for a set delay of seconds.
        # This is to avoid dictionary attacks on Xauth passwords.  Default is one second.
        # Set to zero to disable authentication delay.
```

```
        auth_throttle 10 ;

        # ----------------------------------------------------------------------------
        # CISCO VPN client specific features
        # ----------------------------------------------------------------------------

        # Sets the PFS group used in the client proposal (Cisco VPN client only).  Default is 0.
        pfs_group 2 ;

        # Allow the client to save the Xauth password (Cisco VPN client only).  Default is off.
        save_passwd on ;

        # The split dns configuration to send, in quoted domain name strings.
        # This list can be used to describe a list of domain names for which a peer should query a
        # modecfg assigned dns server.  DNS queries for all other domains would be handled locally.
        # (Cisco VPN client only).
        ##split_dns "home.example.com" ;
}

# ----------------------------------------------------------------------------
# LDAP RELATED
# ----------------------------------------------------------------------------

# Defines the parameters that will be used to communicate with an ldap server for xauth authentication.
##ldapcfg
##{
        # The ldap protocol version used to communicate with the server.  The default is 3.
        ##version 3 ;

        # The host name or ip address of the ldap server.  The default is localhost.
        ##host "ldapserver" ;

        # The port that the ldap server is configured to listen on. The default is 389.
        ##port 389;

        # The ldap search base. This option has no default value.
        ##base distinguished name;

        # Use the subtree ldap search scope.  Otherwise, use the one level search scope.  The default is off.
        ##subtree off;

        # The user dn used to optionally bind before performing ldap search operations. If this option is not
        # specified anonymous binds are used.
        ##bind_dn distinguished name;

        # The password used when binding as bind_dn.
        ##bind_pw "password" ;

        # The attribute used to specify a users name in an ldap directory.  For example, if a user dn is
        # "cn=jdoe,dc=my,dc=net" then the attribute would be "cn". The default value is cn.
        ##attr_user attribute name;

        # The attributes used to specify a users network address and subnet mask in an ldap directory.
        # These values are forwarded during mode_cfg negotiation when the conf_source is set to ldap.
        # The default values are racoon-address and racoon-netmask.
        ##attr_addr attribute name;
        ##attr_mask attribute name;

        # The attribute used to specify a group name in an ldap directory.  For example, if a group dn is
        # "cn=users,dc=my,dc=net" then the attribute would be "cn". The default value is cn.
        ##attr_group attribute name;

        # The attribute used to specify group membership in an ldap directory.  The default value is member.
        ##attr_member attribute name;
##}



# ----------------------------------------------------------------------------
# IKE PHASE 2
# ----------------------------------------------------------------------------

# defines the parameters of the IKE phase 2 (IPsec-SA establishment).
# source_id and destination_id are constructed like:
# address address [/ prefix] [[port]] ul_proto
# or
# subnet address [/ prefix] [[port]] ul_proto
# or
# idtype string
#
# An id string should be expressed to match the exact value of an ID payload (source is the local end,
# destination is the remote end).  This is not like a filter rule.  For example, if you define
# 3ffe:501:4819::/48 as source_id 3ffe:501:4819:1000:/64 will not match.
# In the case of a longest prefix (selecting a single host), address instructs to send ID type of ADDRESS while
# subnet instructs to send ID type of SUBNET.  Otherwise, these instructions are identical.
# The group keyword allows an XAuth group membership check to be performed for this sainfo section.  When the
# mode_cfg auth source is set to system or ldap, the XAuth user is verified to be a member of the specified group
# before allowing a matching SA to be negotiated.
```

```
sainfo anonymous
{

        # define the group of Diffie-Hellman exponentiations.  If you do not require PFS then you can omit this
        # directive. Any proposal will be accepted if you do not specify one. group is one of following: modp768,
        # modp1024, modp1536, modp2048, modp3072, modp4096, modp6144, modp8192.  Alternatively you can define 1, 2,
        # 5, 14, 15, 16, 17, or 18 as the DH group number.
        pfs_group modp1024;

        # define how long an IPsec-SA will be used, in timeunits. Any proposal will be accepted, and no
        # attribute(s) will be proposed to the peer if you do not specify it(them). See the proposal_check
        # directive.
        lifetime time 3600 sec;

        # racoon  does not have a list of security protocols to be negotiated.  The list of security protocols are
        # passed by SPD in the kernel.  Therefore you have to define all of the potential algorithms in the phase 2
        # proposals even if there are algorithms which will not be used.  These algorithms are define by using the
        # following three directives, with a single comma as the separator. For algorithms that can take variable-
        # length keys, algorithm names can be followed by a key length, like “lowfish 448” racoon will compute the
        # actual phase 2 proposals by computing the permutation of the specified algorithms, and then combining
        # them with the security protocol specified by the SPD.  For example, if des, 3des, hmac_md5, and hmac_sha1
        # are specified as algorithms, we have four combinations for use with ESP, and two for AH.  Then, based on
        # the SPD settings, racoon will construct the actual proposals.  If the SPD entry asks for ESP only, there
        # will be 4 proposals. If it asks for both AH and ESP, there will be 8 proposals.  Note that the kernel may
        # not support the algorithm you have specified. des, 3des, des_iv64, des_iv32, rc5, rc4, idea, 3idea,
        # cast128, blowfish, null_enc, twofish, rijndael, aes, camellia (used with ESP).
        encryption_algorithm 3des;

        # des, 3des, des_iv64, des_iv32, hmac_md5, hmac_sha1, hmac_sha256, hmac_sha384, hmac_sha512,
        # non_auth (used with ESP authentication and AH)
        authentication_algorithm hmac_sha1;

        # deflate (used with IPComp)
        compression_algorithm deflate;
}
```

Configuration files – Roadwarrior, gateway1, setkey.conf

```
# This is a sample setkey.conf
# First of all flush the SPD database
flush;
spdflush;
```