strongSwan

The new IKEv2 VPN Solution

Andreas Steffen
andreas.steffen@strongswan.org

Institute for Internet Technologies and Applications
Hochschule für Technik Rapperswil, Switzerland

strongSwan is a complete IPsec-based VPN solution supporting both the traditional IKEv1 as well as the new IKEv2 key exchange protocols. Using practical examples we will present the novel features made possible by IKEv2, among them mixed-mode authentication with the VPN gateway presenting an X.509 certificate and the clients using either pre-shared secrets or one of the various EAP authentication methods (e.g. the SIM-card based EAP-SIM or EAP-AKA methods popular in mobile environments). Other goodies include fast VPN connection setup, built-in NAT traversal and dead peer detection, automatic subnet narrowing, as well as the convenient new IKE configuration payload that can be used to transfer a whole set of network attributes like virtual IP addresses and internal DNS information). We will also give an outlook on our forthcoming "Peer-to-Peer NAT-Traversal for IPsec" Internet draft which proposes an innovative IKEv2 protocol extension to establish IPsec tunnels in double-NAT situations by using UDP end point discovery and hole punching through Network Address Translators assisted by an IKEv2 mediation server.

VPNs revisited

IIllustration 1 shows the two most common uses for a Virtual Private Network. Often a VPN connects two geographically separated sites over the public Internet by means of a cryptographically secured communications link. In our example the hosts 10.1.0.5 and 10.2.0.3 in the two subnets 10.1.0.0/16 and 10.2.0.0/16, respectively, are not aware that their respective VPN gateways with external IP addresses 11.22.33.44 and 55.66.77.88 tunnel all traffic between the subnets by encrypting them according to the IPsec Encapsulated Security Payload (ESP) RFC 4303 standard. We call this application a *site-to-site tunnel*. The second, more challenging use is a *remote access tunnel*, where so-called *road warriors* with dynamical and therefore a priori unknown IP addresses connect to a security gateway in order to access the protected subnet behind it. We will treat this interesting case in more detail later on.

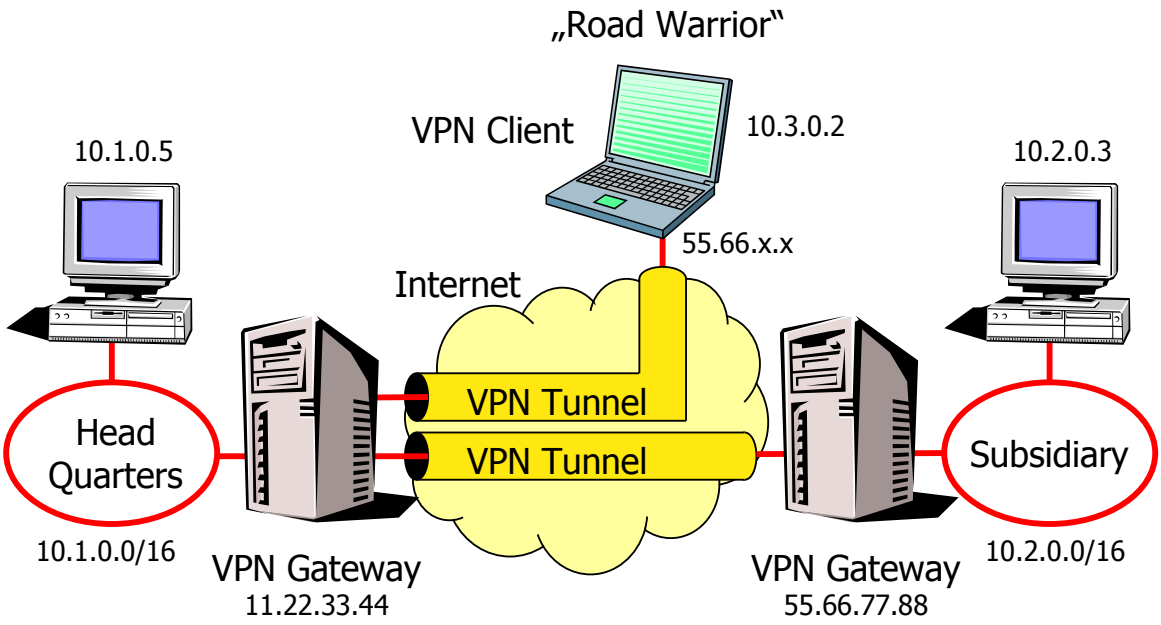

Illustration 1: Site-to-Site and Remote Access Virtual Private Networks

The FreeS/WAN Genealogy

strongSwan is a straight successor of the famous FreeS/WAN project started by John Gilmore in 1999 with the goal of automatically securing a large share of the Internet traffic. His *Opportunistic Encryption* scheme intended to use raw RSA authentication keys stored in the global Domain Name System (DNS).

Unfortunately the restriction to raw RSA keys did not allow FreeS/WAN to inter-operate with a multitude of VPN clients and gateways that were using X.509 certificates. Therefore in the year 2000 the author started to contribute the *X.509 Patch* which added full X.509 certificate and PKCS#11 smart card support to the FreeS/WAN source code. Due to political reasons the X.509 patch never got officially merged into FreeS/WAN but around 2002 Ken Bantoft integrated several add-on patches (X.509, NAT-Traversal, Dead Peer Detection, alternative encryption algorithms, etc.) with FreeS/WAN and put this augmented distribution on the Internet. His *Super FreeS/WAN* quickly became very popular and found its way into a couple of Open Source firewall projects, among them IPCop. Ken Bantoft maintained Super FreeS/WAN until recently under the name of *Openswan 1.x*.

Whereas the original FreeS/WAN 1.x ran on Linux 2.0, 2.2, and 2.4 kernels using its own KLIPS IPsec kernel module, the FreeS/WAN 2.x branch was also able to run on the new Linux 2.6 kernel thanks to Herbert Xu who contributed an XFRM interface which made interaction with the Linux 2.6 kernel's native NETKEY IPsec stack possible. Without the need for KLIPS, FreeS/WAN 2.x could now be built as a pure userland application thus eliminating the tiresome step of recompiling the Linux kernel sources.

In 2004 John Gilmore decided to discontinue the FreeS/WAN project, mainly because he held the view that the main goal of implementing Opportunistic Encryption had been achieved with the final FreeS/WAN 2.0.6 release and thus there was just nothing else to do.

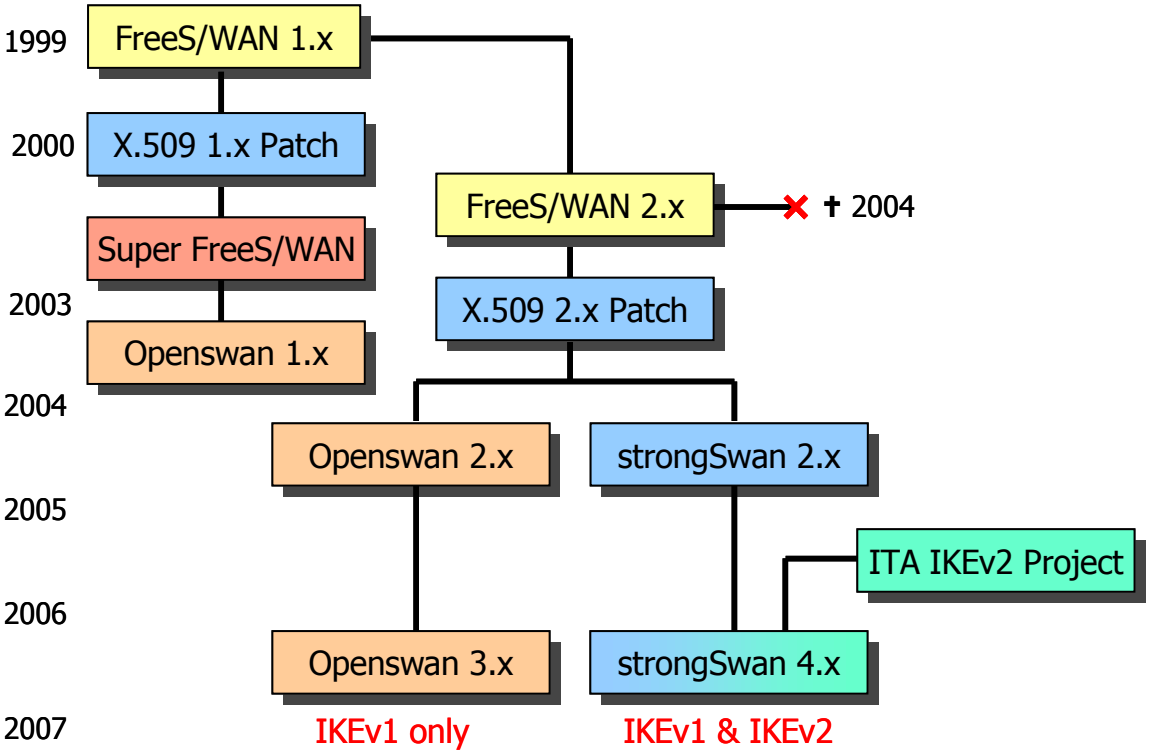

Illustration 2: The FreeS/WAN genealogy

Right after the demise of FreeS/WAN, ex-project leader Michael Richardson teamed up with Ken Bantoft and Paul Wouters to found Xelerance Inc. with the main goal of pursuing the *Openswan* project. Due to various reasons the author decided to fork off a *strongSwan* distribution of his own. Thus Openswan 2.x swerving towards the VPN mainstream e.g. by supporting the potentially insecure IKE aggressive mode and strongSwan 2.x with its focus on strong authentication became the official successors of the FreeS/WAN project (see the family tree in Illustration 2).

In 2005, some six months before the official publication of the IKEv2 RFC 4306, the two HSR students Jan Hutter and Martin Willi approached me with the proposal to design an IKEv2 software architecture based on modern, object-oriented principles and to implement a rapid prototype in the C programming language as part of their diploma thesis at the Institute for Internet Technologies and Applications (ITA). After the successful completion of their prize-winning thesis, Martin Willi decided to stay on at the Institute in order to develop a full-fledged IKEv2 implementation which I now have the honor to present in this paper.

Because we wanted to maintain a maximum compatibility with the existing IKEv1 strongswan-2.x implementation, the well-established *ipsec.conf* and *ipsec.secrets* configuration syntax was kept, with just the exception of some new IKEv2-specific keywords. By bundling the IKEv1 keying daemon *pluto* from the strongswan-2.x branch (having its origins in the FreeS/WAN project) with the modern multi-threaded, object-oriented IKEv2 keying daemon *charon*, we created the strongswan-4.x branch which currently is the only Open Source IPsec implementation offering both IKEv1 and IKEv2 capabilities.

To complete our overview, Openswan-3.x is focusing on a KLIPS IPsec stack for the Linux 2.6 kernel with built-in support of hardware crypto accelerators.

Internet Key Exchange Version 1 (IKEv1)

In this section we give a very concise overview of version 1 of the Internet Key Exchange (IKEv1) protocol; i.e. just enough information to be able to highlight the considerable improvements brought about by the successor protocol IKEv2.

IKEv1 is split into two phases: Phase 1 realized either by IKE Main Mode or IKE Aggressive Mode sets up an ISAKMP security association (SA), comprising mutual peer authentication and the generation of keying material for the secure exchange of IKE messages. Phase 2, implemented by IKE Quick Mode sets up one or several IPsec SAs that produce the ESP keying material required to transmit encrypted and authenticated payload packets.

Illustration 3 shows the Phase 1 IKE Main Mode message exchange effecting a peer authentication based on RSA signatures. IKE uses UDP datagrams with the well-known source and destination port 500.

- In a preliminary message exchange the initiator sends an ISAKMP SA proposal containing a list of cryptographic transforms. The responder selects the first acceptable transform and returns it in the ISAKMP SA response.
- In a second message exchange each peer sends a public Diffie-Hellman factor and a nonce which are used to derive encryption and HMAC keys for all further IKE messages. The nonces protect against replay attacks.
- The third and last exchange of IKE Main Mode is used to confidentially transmit the peer identities, an RSA-signed hash computed over all previous messages and optionally a certificate that can be used by the receiver to verify the RSA signature and with the help of the identity string to authenticate the peer.

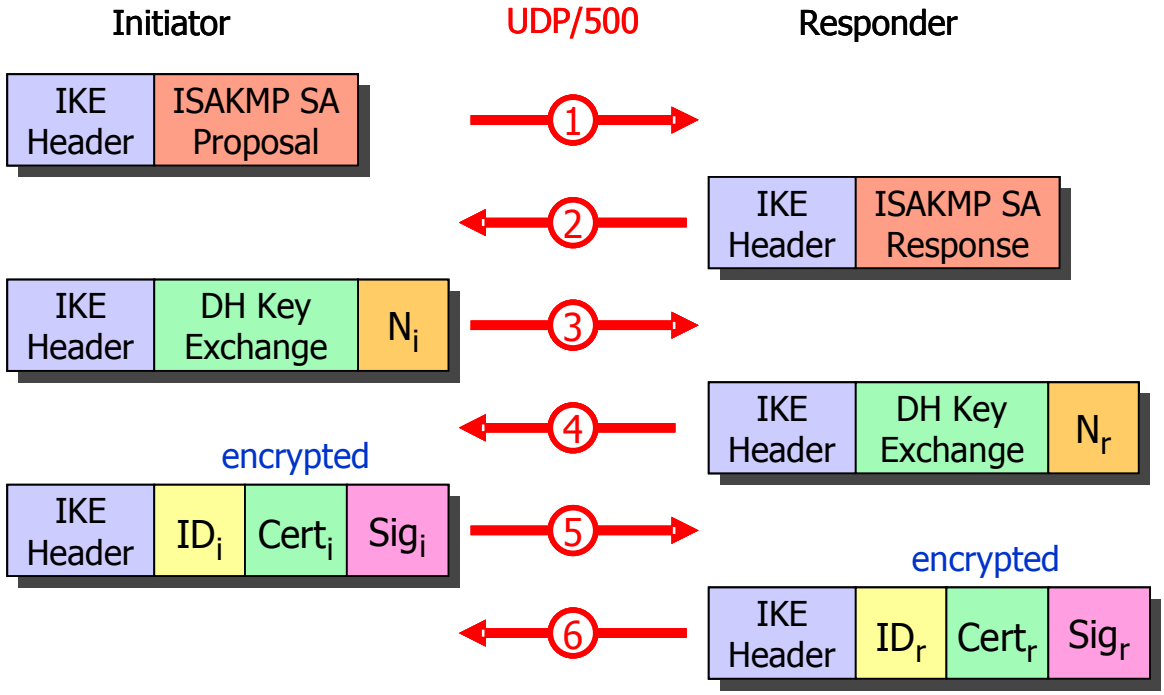

Illustration 3: IKEv1 – IKE Main Mode message exchange

- The ensuing Phase 2 IKE Quick Mode uses three additional messages to exchange the traffic selectors and cryptographic transforms needed for an IPsec SA. In the topology shown in Illustration 1, the traffic selectors for the site-to-site VPN would be 10.2.0.0/16 and 10.1.0.0/16, defining the two subnets that are to be connected by the IPsec tunnel whereas in the remote access case the traffic selectors would be 10.3.0.2/32 and 10.1.0.0/16, connecting the road warrior with the home network.

Thus the establishment of a single IPsec SA using the IKEv1 protocol requires the exchange of a total of **nine** UDP datagrams. This fact alone is already reason enough to develop an improved second generation protocol!

Internet Key Exchange Version 2 (IKEv2)

IKEv2 as defined by RFC 4306 improves considerably upon its predecessor by packing the establishment of a single IPsec SA into a mere **four** UDP datagrams. These messages are shown in Illustration 4.

IKEv2 employs a strict request/response message exchange scheme with the response [besides often also carrying information] always having the function of an acknowledgement. Thus the task of resending messages falls to the initiator, only. In the case of frequent packet loss or network congestion this consistent scheme makes IKEv2 much more stable than IKEv1 where often both sides would start to retransmit messages thought to be lost.

- The IKE_SA_INIT message packs the selection of cryptographic transforms for the IKE SA ($SA1_i$/$SA1_r$), the derivation of a common Diffie-Hellman secret (KE_i/KE_r) and the nonces (N_i/N_r) into a single exchange. Since a Diffie-Hellman public key operation is computationally expensive, the responder can request a cookie if a Denial-of-Service attack is suspected.

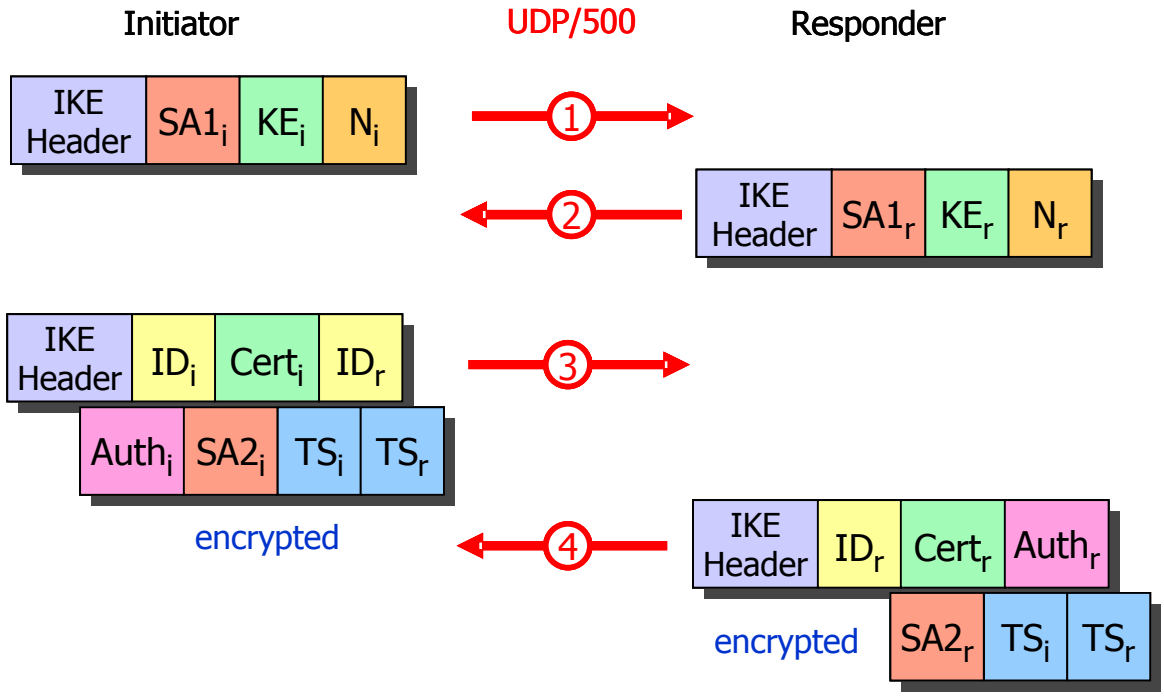

Illustration 4: IKEv2 - IKE_SA_INIT and IKE_AUTH message exchanges

- The ensuing IKE_AUTH message exchange not only authenticates the peers ($AUTH_i$/$AUTH_r$) using pre-shared keys (PSK), RSA signatures, or the extensible authentication protocol (EAP) but also sets up a first so-called Child SA by defining traffic selectors (TS_i/TS_r) and the cryptographic transforms for the IPsec connection ($SA2_i$/$SA2_r$). As part of the authentication each peer also sends his identity string (ID_i/ID_r) and optionally a certificate ($CERT_i$/$CERT_r$). As a novel feature the initiator may request a the responder to take on a specific identity (ID_r) if the peer is known to possess several.
- Multiple Child SAs can be set up by executing the CREATE_CHILD_SA request/response pair shown in Illustration 5 carrying the cryptographic transforms (SA_i/SA_r), a pair of fresh nonces (N_i/N_r), an optional Diffie-Hellman exchange if perfect forward secrecy (PFS) is desired an of course the additional traffic selectors (Ts_i/TS_r). The CREATE_CHILD_SA message exchange is also used for the periodic re-keying of either a Child SA or the IKE SA by including a corresponding notification payload (N).

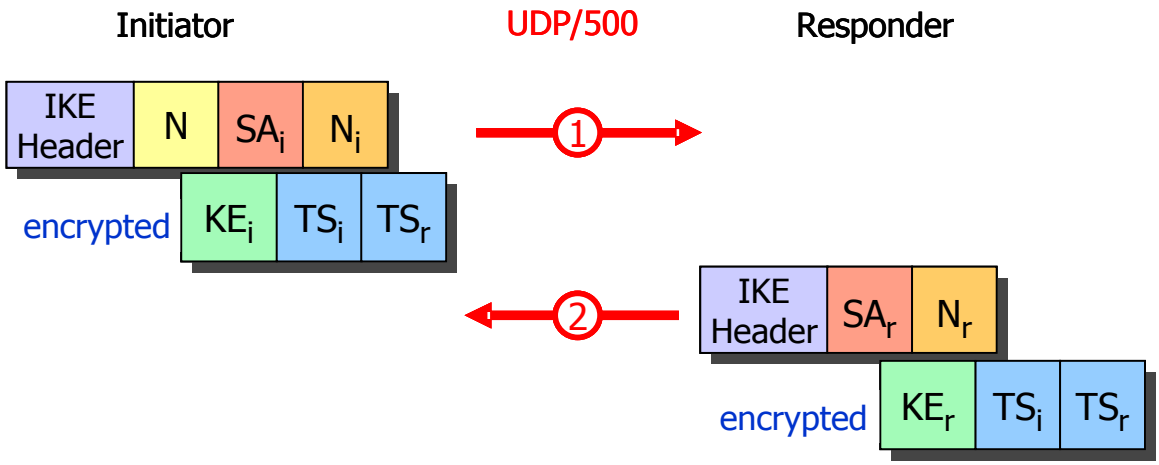

Illustration 5: IKEv2 - CREATE_CHILD SA message exchange

- At any time either peer can send an INFORMATIONAL message which is always acknowledged by a response. As Illustration 6 shows, an INFORMATIONAL request can contain a notify (N), a delete SA (D), or a configuration (CP) payload. Empty INFORMATIONAL exchanges can be used to implement Dead Peer Detection (DPD).

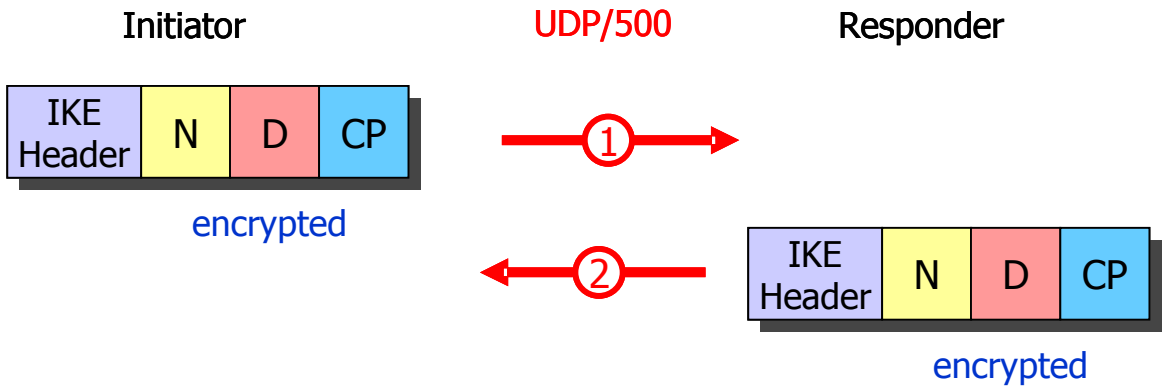

Illustration 6: IKEv2 - INFORMATIONAL message exchange

User-Mode-Linux based Virtual VPN Testbed

We are now going to present two practical IKEv2 scenarios by running them on our flexible User-Mode-Linux based virtual VPN testbed shown in Illustration 7.

The clients *alice* and *venus* are part of the 10.1.0.0/16 subnet hidden behind gateway *moon* whereas client *bob* belongs to the 10.2.0.0/16 subnet located behind gateway *sun*. This particular setup allows the simulation of site-to-site VPN connections.

The 192.168.0.0/24 network models the insecure Internet comprising the outer interfaces of the gateways *moon* and *sun* as well as the two road warriors *carol* and *dave*, both of which can be used in remote access VPN scenarios. The web server *winnetou* functions as a repository for certificate revocation lists (CRLs) and as a responder for the on-line certificate status protocol (OCSP).

The gateways *moon* and *sun* can be configured as NAT routers thus allowing the simulation of IPsec NAT traversal scenarios. In a single NAT topology the VPN clients *alice* and *venus* sitting behind the NAT router *moon* set up a tunnel to the VPN gateway *sun* in order to reach the subnet 10.2.0.0/16 behind it.

By additionally configuring a port forwarding rule for the UDP destination ports 500 and 4500 on gateway *sun*, all IKE and UDP-encapsulated ESP traffic is forwarded to VPN client *bob*, thus creating a double NAT situation that can still be handled by the standard NAT traversal capabilities of the IKEv2 protocol.

If both gateways *moon* and *sun* act as source NAT routers then the VPN clients behind them are not reachable under the standard IKE ports any more and the new peer-to-peer NAT traversal protocol extension described at the end of this paper must be applied in order to be able to establish an IPsec tunnel.

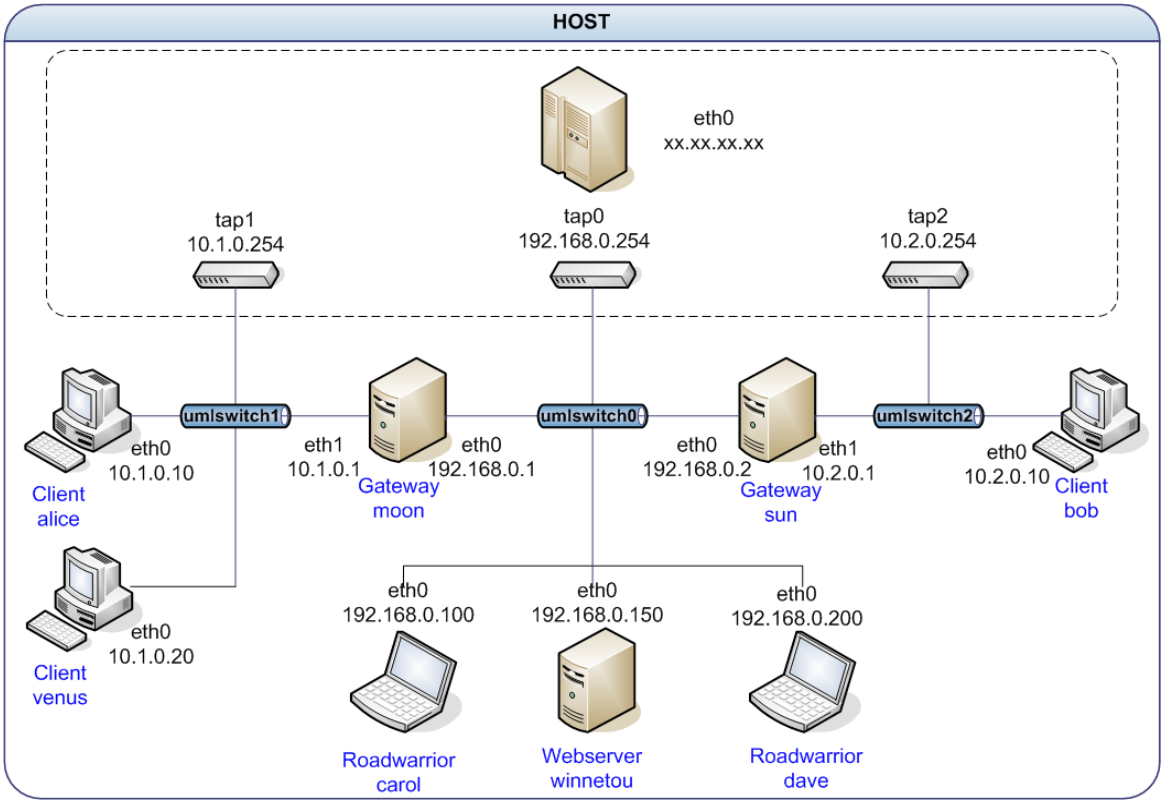

Illustration 7: User-Mode-Linux based virtual VPN testbed

Typical Road Warrior Scenario

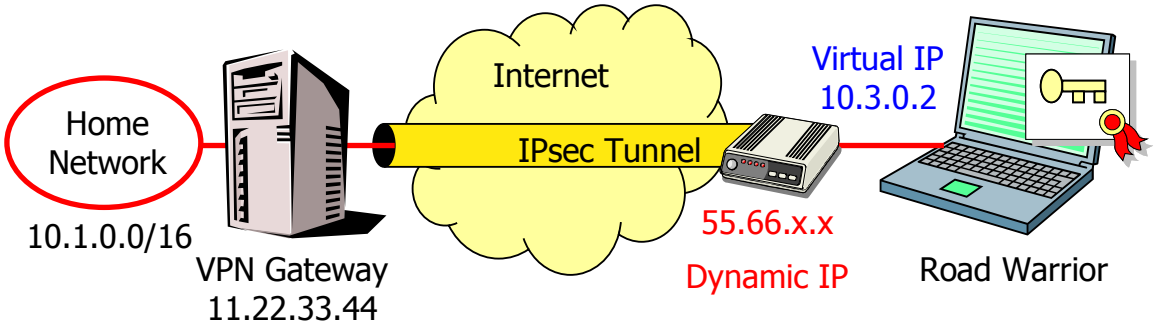

Illustration 8: VPN road warrior remote access case

Illustration 8 shows a typical road warrior scenario which has the following special properties:

- A road warrior usually possesses a dynamic IP address assigned by the current local ISP. Therefore the IP address doesn't carry any information about the peer at all and should not be used as the peer ID.
- Authentication should preferably be based on RSA signatures verifiable by means of X.509 certificates. With IKEv1 this was the only way to use IKE Main Mode with dynamic peer IP addresses. With IKEv2 it has become possible to use mixed RSA/PSK or RSA/EAP authentication modes, though, as we will show later on.
- The VPN gateway should assign to each road warrior a distinct virtual IP address taken from a remote access address pool, to be used as a source address within the IPsec tunnel. This guarantees that return traffic from the home network is reliably routed to VPN gateway and then tunneled back to the road warrior.
- Road warriors are often hidden behind NAT-routers. Thus NAT traversal for IKE and IPsec traffic is a must. The strongSwan IKEv2 implementation automatically takes care of NAT discovery, port floating to NAT-T port 4500 and subsequent UDP encapsulation of ESP packets.

IKEv2 Remote Access with X.509 Certificates

Using the VPN testbed shown in Illustration 7 we want to set up an IKEv2 road warrior scenario consisting of the two remote access clients *carol* and *dave* setting up an IPsec tunnel to VPN gateway *moon* in order to reach the subnet 10.1.0.0/16 behind it. Illustration 9 depicts the configuration files for *carol* on the left hand side and for *moon* on the right hand side. Since mutual authentication is to be based on RSA signatures, both *ipsec.secrets* files contain the path to an RSA private key file located in the */etc/ipsec.d/private/* directory. *carol*'s key file is still protected by a 3DES transport pass phrase which can be appended to the *ipsec.secrets* filename entry so that the RSA key is automatically decrypted during the strongSwan startup.

```
#ipsec.secrets for roadwarrior carol

: RSA carolKey.pem "nH5ZQEWtku0RJEZ6"
```

```
#ipsec.secrets for gateway moon

: RSA moonKey.pem
```

```
#ipsec.conf for roadwarrior carol

conn home
      keyexchange=ikev2
      left=%defaultroute
      leftsourceip=%config
      leftcert=carolCert.pem
      leftid=carol@strongswan.org
      leftfirewall=yes
      right=192.168.0.1
      rightid=@moon.strongswan.org
      rightsubnet=10.1.0.0/16
      auto=start
```

```
#ipsec.conf for gateway moon

conn %default
      keyexchange=ikev2
      left=%defaultroute
      leftcert=moonCert.pem
      leftid=@moon.strongswan.org
      leftfirewall=yes
      right=%any
      auto=add

conn rw-carol
      rightid=carol@strongswan.org
      rightsourceip=10.3.0.1
      leftsubnet=10.1.0.0/24
      lefthostaccess=yes

conn rw-dave
      rightid=dave@strongswan.org
      rightsourceip=10.3.0.2
      leftsubnet=10.1.0.20/32
```

Illustration 9: Configuration files for IKEv2 remote access scenario

Let us first take a look at *carol*'s *ipsec.conf* configuration file. The new IKEv2 protocol is selected by setting

`keyexchange=ikev2`

the default being `keyexchange=ikev1`. The next line

`left=%defaultroute`

makes the current IP address of the road warrior's network interface to the outer address of the IPsec tunnel. The use of the `%defaultroute` wild card is recommended with network address that are dynamically assigned by an ISP and that change frequently as is often the case with remote access clients. Dating back to FreeS/WAN *left* and *right* can be used interchangeably but it is good practice to designate the *local* side by *left* and the *remote* side by *right*.

`leftsourceip=%config`

will force the road warrior to prompt the VPN gateway for a virtual IP address that can then be used as a source address inside the IPsec tunnel.

`leftcert=carolCert.pem`

designates the path to *carol*'s X.509 certificate that is going to be sent to *moon* in the CERT payload of the IKE_AUTH message.

`leftid=carol@strongswan.org`

selects *carol*'s identity that will go into the ID_i payload and that must be certified by a subjectAltName in *carol*'s certificate.

`leftfirewall=yes`

activates the automatic insertion and deletion of iptables rules that let pass tunneled traffic.

`right=192.168.0.1`

is the IP adress of VPN gateway moon.

```
rightid=@moon.strongswan.org
```

is the expected identity of VPN gateway *moon* which will be sent in the ID_r payload.

```
rightsubnet=10.1.0.0/16
```

is the subnet behind VPN gateway moon that *carol* wants to reach. And finally

```
auto=start
```

causes the connection home to be started as soon as the keying daemon is up and running. Illustration 10 is an excerpt from *carol*'s syslog which shows the IKE_SA_INIT and IKE_AUTH message exchanges defined in Illustration 4.

```
05[ENC] generating IKE_SA_INIT request [SA No KE N N]
05[NET] sending packet: from 192.168.0.100[500] to 192.168.0.1[500]
06[NET] received packet: from 192.168.0.1[500] to 192.168.0.100[500]
06[ENC] parsed IKE_SA_INIT response [SA No KE N N CERTREQ]
06[ENC] generating IKE_AUTH request [IDi CERTREQ CERT IDr AUTH CP SA TSi TSr]
06[NET] sending packet: from 192.168.0.100[500] to 192.168.0.1[500]
07[NET] received packet: from 192.168.0.1[500] to 192.168.0.100[500]
07[ENC] parsed IKE_AUTH response [IDr CERT AUTH CP SA TSi TSr]
07[IKE] installing new virtual IP 10.3.0.1
07[AUD] established CHILD_SA successfully
```

Illustration 10: Roadwarrior carol as initiator

On the gateway side we first define some default values that will be valid for all subsequent connection definitions, among them

```
auto=add
```
and
```
right=%any
```

which means that *moon* will wait passively as a responder for the road warriors to initiate the IKEv2 connection setup originating from an arbitrary IP address. Next follow the two connection definitions *rw-carol* and *rw-dave* with some host-specific configurations such as

```
rightsourceip=10.3.0.1
rightsourceip=10.3.0.2
```

which define the virtual IPs to be assigned to *carol* and *dave*, respectively. Illustration 11 shows the syslog of responder *moon* which corresponds to *carol*'s initiator log entries.

```
05[NET] received packet: from 192.168.0.100[500] to 192.168.0.1[500]
05[ENC] parsed IKE_SA_INIT request [SA No KE N N]
05[ENC] generating IKE_SA_INIT response [SA No KE N N CERTREQ]
05[NET] sending packet: from 192.168.0.1[500] to 192.168.0.100[500]
06[NET] received packet: from 192.168.0.100[500] to 192.168.0.1[500]
06[ENC] parsed IKE_AUTH request [IDi CERTREQ CERT IDr AUTH CP SA TSi TSr]
06[IKE] peer requested virtual IP %any
06[IKE] assigning virtual IP 10.3.0.1 to peer
06[AUD] established CHILD_SA successfully
06[ENC] generating IKE_AUTH response [IDr CERT AUTH CP SA TSi TSr]
06[NET] sending packet: from 192.168.0.1[500] to 192.168.0.100[500]
```

Illustration 11: VPN gateway moon as responder

IKEv2 Narrowing of Traffic Selectors

A novel feature introduced by the IKEv2 protocol is the automatic *narrowing* of traffic selectors as shown in Illustration 9 where *carol* defines

```
rightsubnet=10.1.0.0/16
```

and inserts this desired subnet into the TS_r traffic selector payload of the IKE_AUTH request but *moon* restricts access to the local subnet to

```
leftsubnet=10.1.0.0/24
```

communicating this narrowing via the TS_r payload in the IKE_AUTH reply, causing the Child SA to be installed with the narrower subnet definition as can easily be seen from the output of *carol*'s ipsec statusall listed in Illustration 12.

```
carol> ipsec statusall

Connections:
  home: 192.168.0.100[carol@strongswan.org]...192.168.0.1[@moon.strongswan.org]
  home: dynamic/32 === 10.1.0.0/16
Security Associations:
  home[1]: ESTABLISHED, 192.168.0.100[carol@strongswan.org]...
                        192.168.0.1[@moon.strongswan.org]
  home[1]: IKE SPIs: 0x7b7be8689f4338ed_i* 0x48a20dbd8a3ae8eb_r,
                        reauthentication in 56 minutes
  home{1}: INSTALLED, TUNNEL, ESP SPIs: 0xcb89fc54_i 0xc28935c3_o
  home{1}: AES_CBC-128/HMAC_SHA1_96, rekeying in 14 minutes, last use: 5s_i 5s_o
  home{1}: 10.3.0.1/32 === 10.1.0.0/24
```

Illustration 12: Narrowing the traffic selectors

The narrowing mechanism is a very convenient feature because the peer does not have to know a priori to which remote subnet he has access to. Taken to the extreme, a VPN client could request

```
rightsubnet=0.0.0.0/0
```

and the peer would reply with the actual traffic selectors that are available for the given host or user based on the particular access control rights.

IKEv2 Configuration Payload

Another new feature is the IKEv2 configuration payload (CP) which is closely modeled after the expired IKE Mode Config Internet draft `<draft-dukes-ike-mode-cfg-02.txt>` which is nevertheless being actively used by Cisco Systems and many other VPN vendors in current IKEv1-based products. Illustration 10 and Illustration 11 both show that *carol* is sending a CP payload in order to request a virtual IP address and *moon* is sending back the pre-assigned address in a CP reply payload.

How is the virtual IP address actually configured on the VPN client *carol*? The output of the Linux system commands `ip addr list` and `ip route list` shown in Illustration 13 gives some insights: The IKEv2 daemon first creates an alias for the virtual IP address 10.3.0.1 and binds it to the physical IPsec interface eth0. It then creates a route entry which forces all IP packets destined for the 10.1.0.0/24 subnet to take on the virtual IP as a source address. Thus all

```
carol> ip addr list dev eth0
eth0: inet 192.168.0.100/24 brd 192.168.0.255 scope global eth0
      inet 10.3.0.1/32 scope global eth0

carol> ip route list
10.1.0.0/24 dev eth0 proto static src 10.3.0.1
```

Illustration 13: Virtual IP assigned to carol

tunneled plaintext packets originate from 10.3.0.1 whereas the encrypted ESP packets originate from the physical network address 192.168.0.100.

Another special feature can be observed on gateway *moon*. As Illustration 14 shows, all IP packets destined for *carol*'s virtual IP address 10.3.0.1 and originating from the gateway itself, automatically assume the source address 10.1.0.1 belonging to the internal eth1 interface and are therefore tunneled to *carol* because there is a successful match against the traffic selectors installed by the Child SA. Without the inserted route, the source address would by default be equivalent to the IP address 192.168.0.1 of the outer eth0 interface and the packets would not go through the tunnel but be sent in the clear. This source address mechanism is automatically activated if one of the network interfaces on a gateway forms part of a subnet that is being tunneled.

```
moon> ip addr list
eth0: inet 192.168.0.1/24 brd 192.168.0.255 scope global eth0
eth1: inet 10.1.0.1/16 brd 10.1.255.255 scope global eth1

moon> ip route list
10.3.0.1 dev eth0 proto static src 10.1.0.1
```

Illustration 14: Using internal interface as source IP

Since it is often not desirable that peers should have access to the VPN gateway itself, although the inner gateway interface forms part of the destination subnet, access must be given explicitly with the statement

```
lefthostaccess=yes
```

as is the case for *carol* in the rw-carol connection definition of Illustration 9.

Full Integration with Linux Netfilter Firewall

With the help of the `leftfirewall=yes` setting which automatically inserts bi-directional FORWARD rules on VPN gateways or an INPUT and OUTPUT rule on single hosts, and the `lefthostaccess=yes` directive which adds an INPUT and OUTPUT rule to reach a gateway itself, strongSwan can open a Linux netfilter based firewall configured with a default DROP or REJECT policy to the tunneled IPsec traffic according to the negotiated traffic selectors. Illustration 15 shows the firewall settings on the VPN gateway *moon* after the two road warriors *carol* and *dave* had set up their tunnels and pinged the hosts *alice* (10.1.0.10) and *venus* (10.1.0.20) successfully.

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target prot in  out  source       destination
 0      0  ACCEPT all eth0  *   10.3.0.1      10.1.0.0/24
                               policy match dir in pol  ipsec reqid 1 proto 50
 2    304  ACCEPT esp eth0  *   0.0.0.0/0     0.0.0.0/0
 4   4720  ACCEPT udp eth0  *   0.0.0.0/0     0.0.0.0/0 udp spt:500 dpt:500

Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target prot in  out  source        destination
 1     84  ACCEPT all eth0  *   10.3.0.2      10.1.0.20
                               policy match dir in pol  ipsec reqid 2 proto 50
 1     84  ACCEPT all  *   eth0 10.1.0.20     10.3.0.2
                               policy match dir out pol ipsec reqid 2 proto 50
 1     84  ACCEPT all eth0  *   10.3.0.1      10.1.0.0/24
                               policy match dir in pol  ipsec reqid 1 proto 50
 1     84  ACCEPT all  * eth0   10.1.0.0/24   10.3.0.1
                               policy match dir out pol ipsec reqid 1 proto 50

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target prot in  out  source        destination
 0      0  ACCEPT all  *   eth0 10.1.0.0/24   10.3.0.1
                               policy match dir out pol ipsec reqid 1 proto 50
 2    304  ACCEPT esp  *   eth0 0.0.0.0/0     0.0.0.0/0
 4   4026  ACCEPT udp  *   eth0 0.0.0.0/0     0.0.0.0/0 udp spt:500 dpt:500
```

Illustration 15: Full integration with Linux netfilter firewall

netfilter's IPsec policy matching capability developed by Patrick McHardy is used to make sure that only traffic coming out of an IPsec tunnel or going into a tunnel can qualify for an inserted iptables rule. Each rule is bound to a tunnel through its reqid (the number listed for each security association by ipsec status). The INPUT and OUTPUT rules for the IKE port and the ESP protocols must be configured statically.

Online Certificate Status Protocol (OCSP)

When working with X.509 certificates it is of utmost importance to be able to revoke compromised certificates in a timely fashion. This can be done either by frequently publishing a certificate revocation list (CRL) or by running one or several OCSP servers. Information on available CRL distribution points or URIs of OCSP servers can be gathered either from special extensions contained in X.509v3 client and host certificates or the URIs can be manually configured in *ipsec.conf* with the help of a special *ca* section. The command *ipsec listcainfos* is used to get an overview on the currently available information. A typical output containing one CRL and one OCSP URI is shown below:

```
moon> ipsec listcainfos

Apr 15 14:58:27 2007
    authname: 'C=CH, O=Linux strongSwan, CN=strongSwan Root CA'
    authkey:   5d:a7:dd:70:06:51:32:7e:e7:b6:6d:b3:b5:e5:e0:60:ea:2e:4d:ef
    crluris:  'http://crl.strongswan.org/strongswan.crl'
    ocspuris: 'http://ocsp.strongswan.org:8880'
```

Illustration 16: List of CRL and OCSP URIs

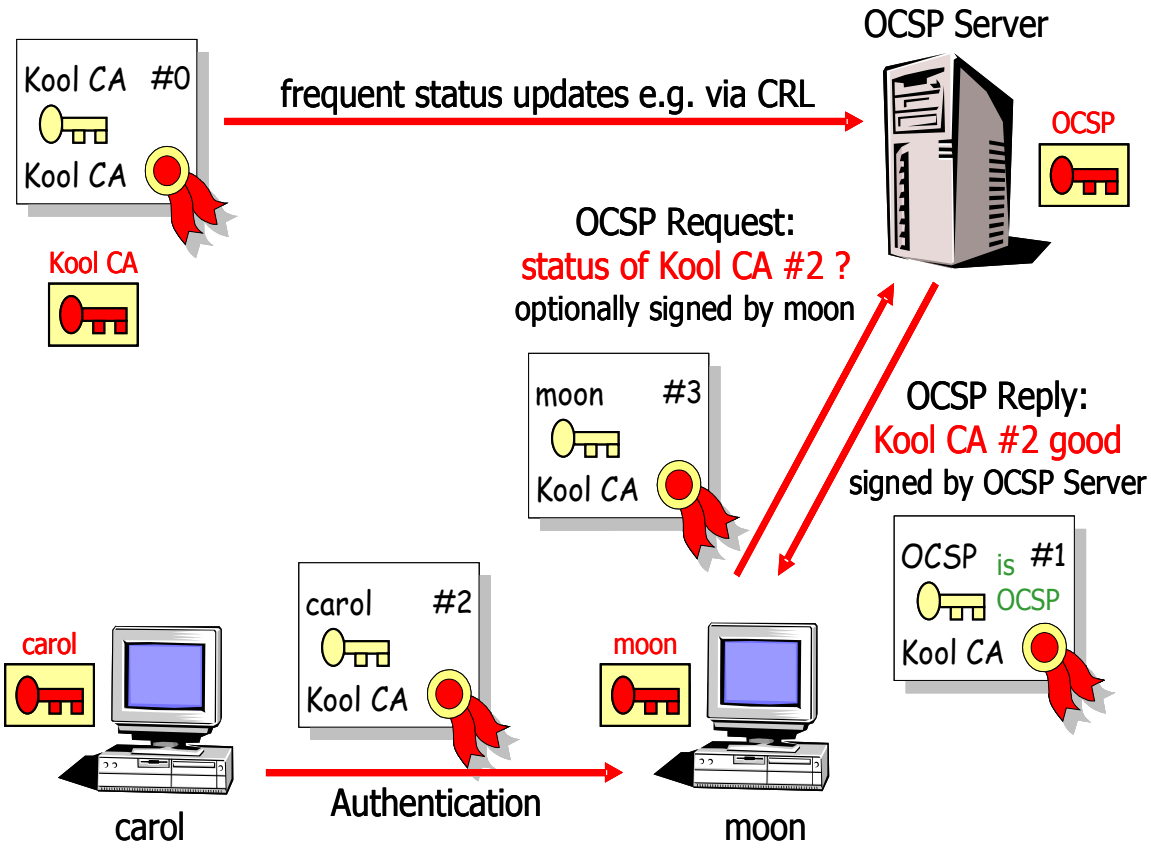

Illustration 17: Online Certificate Status Protocol (OCSP)

Illustration 17 depicts a typical OCSP message flow where the Kool CA is delegating the OCSP service to the http server *ocsp* by issuing a certificate containing an OCSPSigning extended key usage flag to the server. If gateway *moon* is receiving a user certificate signed by the Kool CA from road warrior *carol* and the current certificate status is not known then *moon* will send an OCSP request containing the name of the certification authority and the serial number of *carol*'s certificate to the OCSP server and will receive a signed reply containing one of the possible status values: *good*, *revoked*, or *unknown* as can be seen from the syslog shown in Illustration 18

```
05[LIB] ocsp status is not in cache
05[LIB] sending http post request to 'http://ocsp.strongswan.org:8880'...
05[LIB] received valid http response
05[LIB] received ocsp signer certificate is trusted
05[CFG] certificate is good
```

Illustration 18: Log entry showing http-based OCSP fetching

```
moon> ipsec listocsp
    authname: 'C=CH, O=Linux strongSwan, CN=strongSwan Root CA'
    authkey:   5d:a7:dd:70:06:51:32:7e:e7:b6:6d:b3:b5:e5:e0:60:ea:2e:4d:ef

Apr 15 14:58:30 2007, until Apr 15 15:03:30 2007, ok (expires in 4 minutes)
    serial: 13, good
```

Illustration 19: Cached OCSP reply

IKEv2 Mixed PSK/RSA Authentication

The IKEv2 protocol allows mixed authentication modes where a VPN gateway possesses an X.509 certificate plus a matching private RSA key doing digital signatures whereas the VPN clients use personalized pre-shared keys. A corresponding configuration is shown in Illustration 20.

```
#ipsec.secrets for roadwarrior carol

carol@strongswan.org : \
      PSK "FpZAZqEN6Ti9sqt4ZP5EWcqx"
```

```
#ipsec.conf for roadwarrior carol

conn home
      keyexchange=ikev2
      authby=psk
      left=%defaultroute
      leftid=carol@strongswan.org
      leftfirewall=yes
      right=192.168.0.1
      rightid=@moon.strongswan.org
      rightsubnet=10.1.0.0/16
      auto=start
```

```
#ipsec.secrets for gateway moon

: RSA moonKey.pem

carol@strongswan.org : \
      PSK "FpZAZqEN6Ti9sqt4ZP5EWcqx"

dave@strongswan.org : \
      PSK "jVzONCF02ncsgiS1mIXeqhGN"
```

```
#ipsec.conf for gateway moon

conn rw
      keyexchange=ikev2
      authby=rsasig
      left=%defaultroute
      leftsubnet=10.1.0.0/16
      leftcert=moonCert.pem
      leftid=@moon.strongswan.org
      leftfirewall=yes
      right=%any
      auto=add
```

Illustration 20: IKEv2 Mixed PSK/RSA Authentication

The authby parameter defines the type of authentication the local peer is going to use for himself. Thus the road warrior *carol* has

```
authby=psk
```

and stores a pre-shared key in *ipsec.secrets* which is also known by *moon*. *moon* itself possesses a private RSA key and therefore uses the default authentication mode

```
authby=rsasig
```

IKEv2 EAP Authentication

A SIM card (GSM/GPRS) or a USIM card (UMTS/CDMA2000) is the classical user credential in a mobile communications environment. But increasingly unlicensed public and private WLANs are mushrooming and the straightforward solution for safeguarding the authentication process and subsequent Internet access is to use an IPsec tunnel established by the IKEv2 protocol in combination with a mixed EAP/RSA authentication process. Actually there is a 3GPP standard mandating this procedure, with the consequence that the majority of todays IKEv2 applications are in the mobile communications area. Illustration 21 shows the typical situation where a mobile station (MS) sets up an IPsec tunnel to a security gateway (SEGW) co-located with the generic access network controller (GANC).

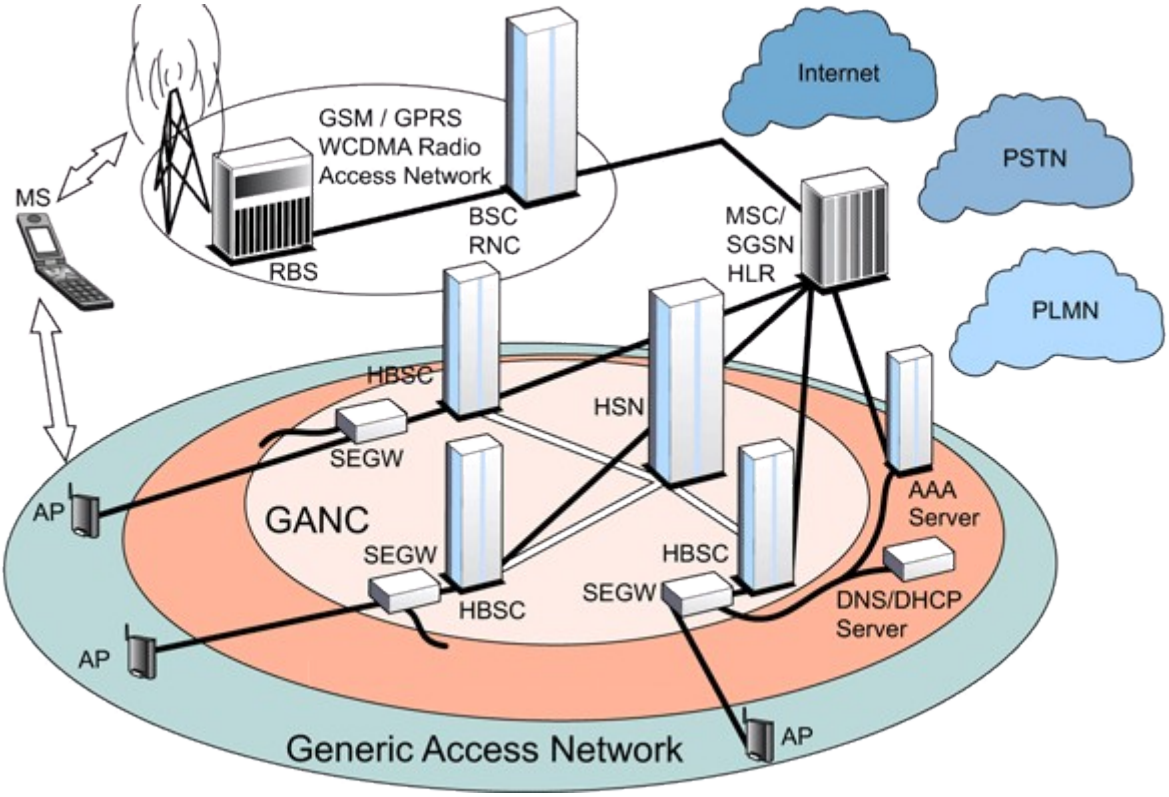

Illustration 21: IKEv2 EAP-SIM or EAP-AKA authentication in mobile applications

The strongSwan project team recently developed both an EAP-SIM and EAP-AKA authentication module for two major mobile communications equipment manufacturers.

IKEv2 Interoperability

The strongSwan team (Martin Willi and Andreas Steffen) participated in the third IKEv2 interoperability test organized by ICSAlabs which took place from March 5-9 2007 in Orlando, Florida. The strongSwan software successfully interoperated with IKEv2 products from

> Alcatel-Lucent, Certicom, CheckPoint, Cisco, Furukawa, Ixia, Juniper, Nokia, SafeNet, Secure Computing, and SonicWall.

Illustration 22 conveys some impressions from the event, showing Martin Willi in full action testing our code. Our conclusion from the bake-off was that strongSwan can compete quite well with commercial implementations.

Another interesting questions concerns the availability and maturity of Open Source IKEv2 software implementations. Currently there are four of them:

- OpenIKEv2 - http://openikev2.sourceforge.net/
- Racoon2 - http://www.racoon2.wide.ad.jp/
- IKEv2 - http://ikev2.zemris.fer.hr/
- strongSwan - http://www.strongswan.org/

The developers of the OpenIKEv2 project have compiled an IKEv2 feature comparison list which is shown in Illustration 23.

Illustration 22: IKEv2 Interoperability Workshop in Orlando, Florida

The Racoon2 and IKEv2 projects were the first to publish usable code but their activity has been rather low to non-existant for more than a year now. Currently the most active projects are OpenIKEv2 and strongSwan. OpenIKE2 initially had a big head start at the beginning of 2006 but strongSwan gained rapidly over the last year so that today the two projects are about even in respect to the number of implemented features. It is worth mentioning that strongSwan is the only Open Source solution that offers NAT traversal, an absolute must for remote access solutions.

Features	openikev2	racoon2	ikev2	strongSwan
Version	0.93	02/11/2005	1.0	4.1.2
Cookies	Yes	Yes	Yes	Yes
DH group negotiation	Yes	Yes	Yes	Yes
Proposal negotiation	Yes	Yes	Yes	Yes
Traffic selector negotiation	Yes	Yes	Yes	Yes
Narrowing	Yes	No	No	Yes
Preshared-Key Authentication	Yes	Yes	Yes	Yes
Certificate Authentication	Yes	No	No	Yes
Child SA Rekeying	Soft	Soft	Soft	Yes
IKE SA Rekeying	Soft	Soft	Soft	Yes
Child SA Deletion	Yes	Yes	Yes	Yes
IKE SA Deletion	Yes	Yes	Yes	Yes
Configuration Payload (Dynamic Addressing)	Yes	No	No	Yes
NAT Traversal	No	No	No	Yes
EAP Support	Yes	No	No	Yes
Tunnel Mode IPSec	Yes	Yes	Yes	Yes
Transport Mode IPSec	Yes	Yes	Yes	Yes
IPSec Interface	XFRM / PFKEYv2	PFKEYv2	PFKEYv2	XFRM
Perfect Forward Secrecy for CHILD_SAs	Yes	Yes	No	Yes
IPv6 support	Yes	Yes	No	Yes
Different configuration per peer	Yes	Yes	Yes	Yes
Repeated Authentication (RFC4478)	Yes	No	No	Yes (not RFC4478)

Illustration 23: IKEv2 Open Source Software Feature Comparison

Peer-to-Peer NAT-Traversal for IPsec

The two HSR graduates Tobias Brunner and Daniel Röthlisberger developed a peer-to-peer NAT traversal scheme for IPsec as part of their diploma thesis. Tobias Brunner then took on the work to write an Internet draft that is now nearly ready to get published. In a preview we would like to outline the proposed peer-to-peer NAT traversal extension of the IKEv2 protocol.

The problem of setting up a UDP connection between two peers hidden behind two NAT routers is well known from the field of IP telephony where STUN and ICE are used to discover and exchange endpoints and concerted hole punching is employed to surmount stateful inspection firewalls.

In a VoIP environment UDP endpoint discovery is usually achieved by means of a STUN server located in the Internet that is reachable by both peers. We plan to use a mediation server using IKEv2 instead, as shown in Illustration 24.

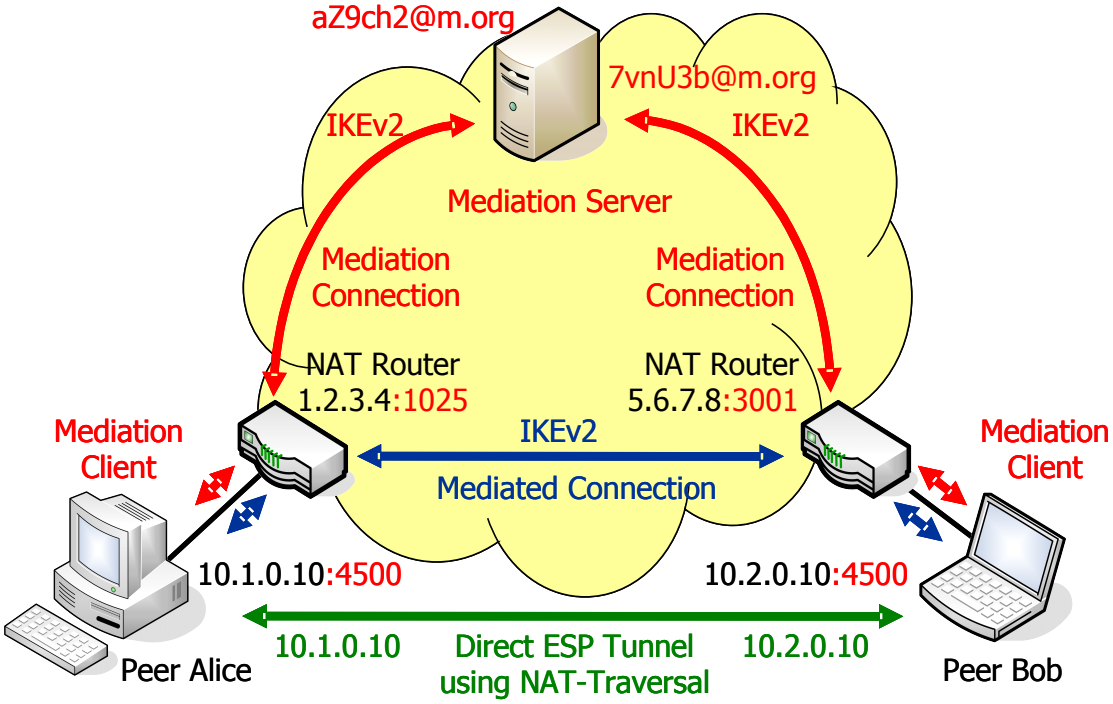

Illustration 24: Peer-to-peer NAT-traversal for IPsec

The two hosts *alice* and *bob* with IP addresses 10.1.0.10 and 10.2.0.10 are sitting behind two NAT routers with external IP addresses 1.2.3.4 and 5.6.7.8, respectively. The peers want to set up a direct IPsec tunnel using the established NAT traversal mechanism of encapsulating ESP packets in UDP datagrams. Unfortunately they cannot achieve this by themselves because neither host is seen from the Internet under the IKE NAT-T port 4500.

Therefore both peers set up a *mediation connection* with a *mediation server*, first. For the mediation connections randomized pseudonyms can be used as IKE peer identities in order to prevent unwanted connection attempts by foreign peers. In our example *alice* sets up an IKE SA with the mediation server using her pseudonym *aZch2@m.org*. As part of our proposed IKEv2 protocol extension no Child SA is created but *aZch2@m.org* can register a request with the mediation server to be alerted when peer *7vnU3b@m.org* comes on-line.

With the help of a new IKEv2 endpoint payload the mediation server tells *alice* under which UDP endpoint she is seen from the Internet (1.2.3.4:1025). When bob sets up his mediation connection in turn, he learns his current endpoint (5.6.7.8:3001) as well.

The mediation server now informs *aZch2@m.org* of *7vnU3b@m.org*'s presence and mediates the mutual exchange of endpoints. a*lice* and *bob* then try to set up a direct IKEv2 connection using their true identities by applying the various hole punching methods described by ICE. The peers might even discover that they are located behind the same NAT router so that no NAT traversal is necessary at all.

Conclusion

We hope to have succeeded in our goal of convincing you of the advantages of the new IKEv2 key exchange protocol and to show that strongSwan is offering a mature and nearly complete Open Source implementation of this emerging standard.

Bibliography

- IETF RFC 4303 *„IP Encapsulating Security Payload”*, 2005
- IETF RFC 4306 *„Internet Key Exchange (IKEv2) Protocol“*, 2005
- IETF RFC 3489 *„STUN - Simple Traversal of UDP Through NATs“, 2003*
- IETF Draft <draft-ietf-mmusic-ice-15.txt> *„ICE - Interactive Connectivity Establishment“*, 2007