# strongSwan VPNs

# scalable and modularized!

**Andreas Steffen**
**andreas.steffen@strongswan.org**

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

*The new strongSwan 4.2 IKEv2 release has been completely modularized. As an alternative to the classical ipsec.conf and ipsec.secrets configuration files, dynamical back end plugins now allow to load configuration data and user credentials from SQLite or MySQL databases. Together with a powerful XML-based web front end, thousands of IPsec tunnels can easily be managed, making strongSwan a truly scalable, industry-grade VPN solution.*

*In addition to these new management features, all encryption and authentication algorithms have been implemented as pluggable modules, thus making it possible to integrate hardware-based crypto-accelerators, use the FIPS 140-2 certified OpenSSL library or attach smart card based secure authentication and storage modules.*

*The strongSwan Mediation Manager is a FastCGI based web interface that writes configuration data and RSA public keys into an SQL database which in turn is accessible by the strongSwan Mediation Server. This leads to a user-friendly solution that makes it possible for two users hidden behind NAT routers each to establish a direct peer-to-peer IPsec tunnel, employing a coordinated UDP hole punching approach.*

## What is strongSwan?

strongSwan is a complete Open Source VPN solution for the Linux operating system. Released under the GNU General Public License (GPLv2) the source code can be freely downloaded from www.strongswan.org. The strongSwan software implements the IKEv1 (RFC 2409) and IKEv2 (RFC 4306) Internet Key Exchange protocols that are needed to set up secure IPsec tunnel connections in an automated way. Illustration 1 shows the the two main application areas:

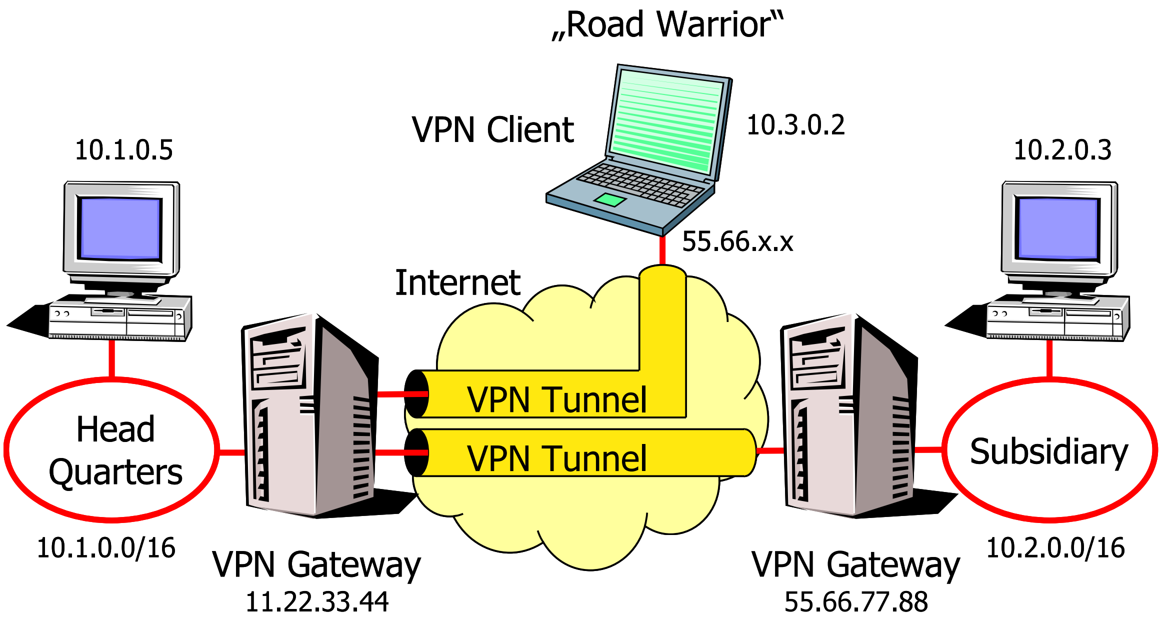


*Illustration 1: Site-to-site and remote-access VPNs*

- *Site-to-site VPNs* that interconnect subnetworks at geographically disjoint locations via the public Internet. Typically the VPN gateways in front of the subnets have static IP addresses.
- *Remote-access VPNs* that allow "road warriors" to connect to their home network from any place on the globe. Typically road warriors have dynamically-allocated IP addresses, usually hide behind NAT routers and often make use of multi-homing network interfaces.

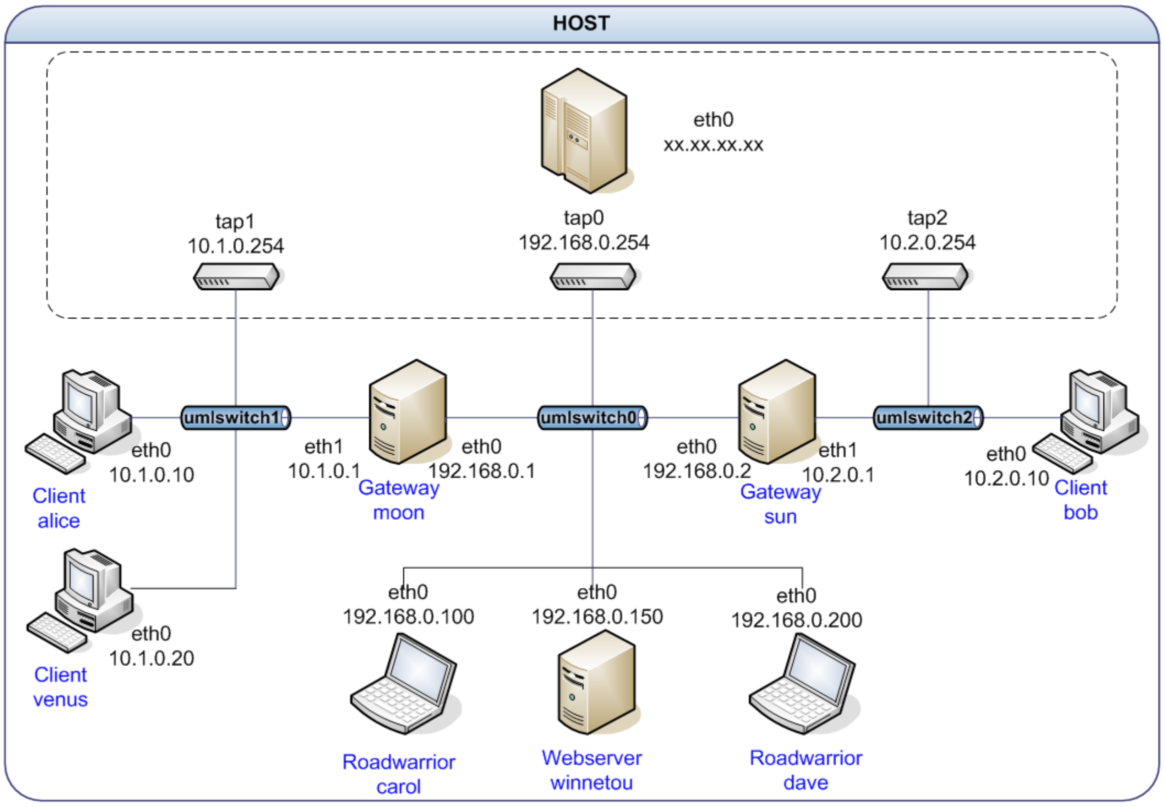


*Illustration 2: User-Mode-Linux based VPN testbed*

The strongSwan developers extensively use the User-Mode-Linux (UML) based virtual testbed shown in Illustration 2 for testing a broad range of VPN scenarios. Both site-to-site and remote-access setups can be simulated. The current test suite employed by our software regression tests comprises 84 IKEv1 and 62 IKEv2 examples based on IPv4 addresses and, due to the increasing deployment of IPv6 technology, a growing number of IPv6 scenarios.

## strongSwan Software Architecture

The main focus of this paper is on the new modular software architecture introduced by the strongSwan 4.2 release. Since strongSwan has a long history going back to the famous FreeS/WAN project started in 1999 (see the FreeS/WAN family tree in Illustration 3), certain design constraints had to be obeyed in order to guarantee the full backward compatibility to previous strongSwan versions. The following two decisive milestones that occurred during the lifetime of the strongSwan project are worth mentioning:

- The strongSwan project was founded by the author in March 2004 as a fork from the FreeS/WAN project (www.freeswan.org) which was officially discontinued by its sponsor John Gilmore with the final freeswan-2.04 version released in April 2004. Both strongSwan and the second fork Openswan initially shared the same code basis including the author's X. 509 patch that added certificate and smartcard support to FreeS/WAN's basic IKEv1 capability. But whereas Openswan rather followed the VPN mainstream by supporting IKE Aggressive Mode, strongSwan focussed on strong certificate and smartcard based authentication mechanisms.

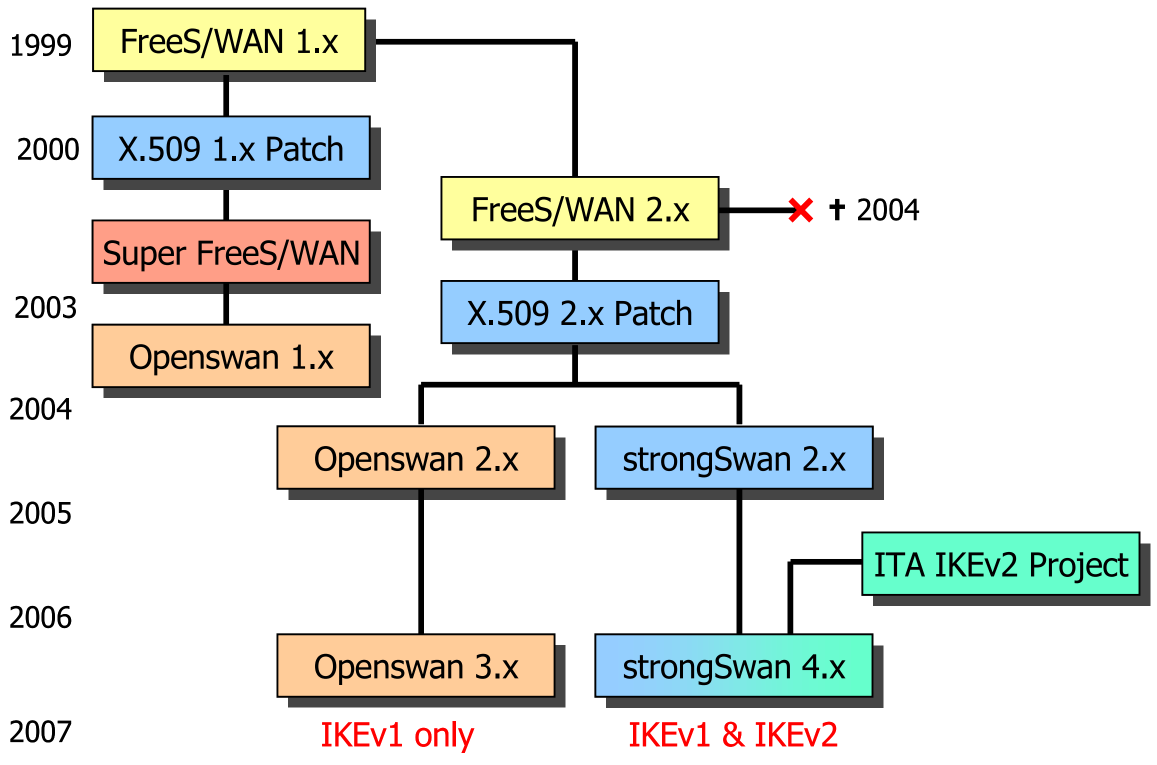


*Illustration 3: The Frees/WAN genealogy*

- The second milestone occurred in May 2006 when the strongSwan 4.x branch incorporating a first partial implementation of the IKEv2 protocol was presented at LinuxTag2006 in Wiesbaden. Whereas the IKEv1 *pluto* daemon inherited from the strongSwan 2.x branch was still heavily based on the original FreeS/WAN code, the IKEv2 *charon* daemon had been developed from scratch by applying modern object-oriented software design principles. In order to achieve full configuration compatibility between IKEv1 and IKEv2 connections, the architectural approach of Illustration 4 was chosen:

  The independent *pluto* and *charon* daemons are both controlled by the *ipsec starter* which parses the *ipsec.conf* configuration file and then transmits the connection information to the daemons via the *whack* and *stroke* socket interfaces, respectively. The *whack* and *stroke* record-based protocols are quite similar and are also spoken by the *ipsec whack* and *ipsec stroke* command line tools which are mainly used to send control and status commands directly to the keying daemons. Typical commands are *up <conn>*, *down <conn>*, *statusall*, *listall*, etc.

  The keying daemons communicate with the native IPsec stack of the Linux 2.6 kernel via a Netlink socket which speaks the XFRM protocol. IPsec SAs can be inserted and deleted and status information on the active tunnels can be retrieved from the kernel which does the actual ESP encryption and decryption work. Thus strongSwan is a pure userland application that interoperates with a standard IPsec-enabled Linux kernel.

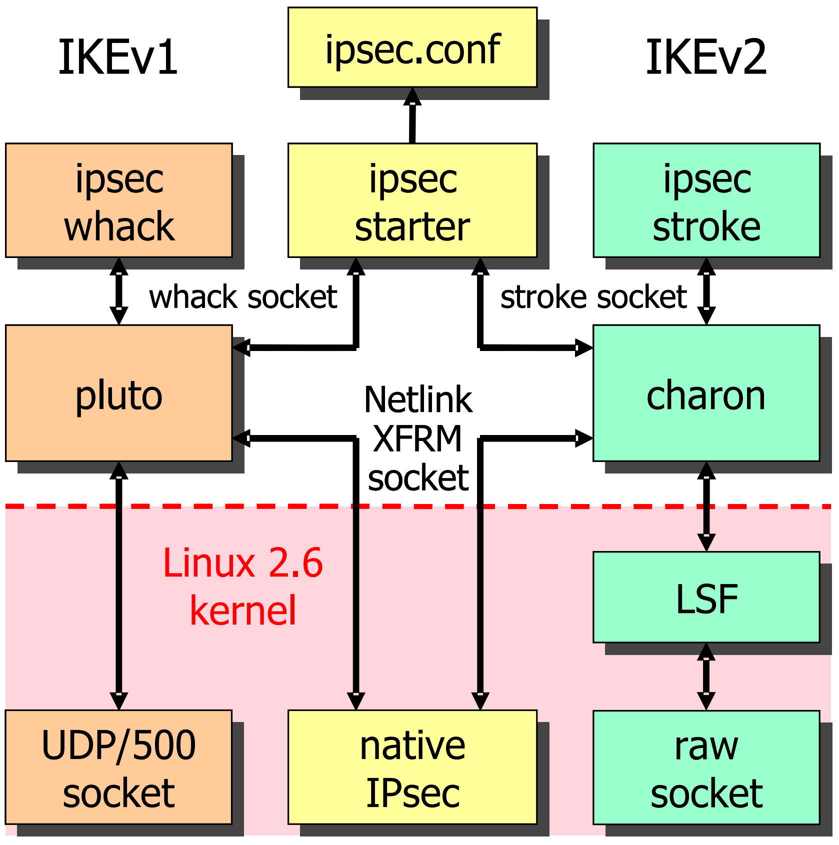


*Illustration 4: The strongSwan IKE daemons*

## Software Architecture of the IKEv2 Daemon

In the following we are going to concentrate on the IKEv2 *charon* daemon only, since no active development is taking place for the “old” IKEv1 *pluto* daemon any more. Fresh IKEv1 functionality might be integrated into *charon* at a future development stage. Illustration 5 shows a block diagram of *charon*'s software architecture. Written completely in the C programming language, the IKEv2 daemon is fully multi-threaded. A total of 16 worker threads concurrently handle a broad set of different tasks. E.g. an IKE packet arriving at the UDP *socket* interface is assigned to an idle *receiver* thread which pre-parses the message and then queues a job with the *schedule*r for further processing. Reply packets are then sent by the *sender* thread to the *socket* interface.

In addition to the UDP socket the following blocks are interfacing with the outside world:

- **backends** retrieve connection definitions
- **credentials** manage Pre-Shared Keys (PSKs), RSA private keys and fetch X.509 certificates and certificate revocation lists (CRLs).
- **bus** is responsible for the signalling of events among the threads and for logging to the syslog, the file system or a database.
- **kernel interface** communicates with the IPsec stack in the Linux 2.6 kernel via the Netlink socket interface using the XFRM protocol.

At the core of the charon daemon is the *IKE SA Manager* which is responsible for the peer authentication based on the presented credentials and sets up IKE_SAs and dependent CHILD_SAs according to the connection definitions.

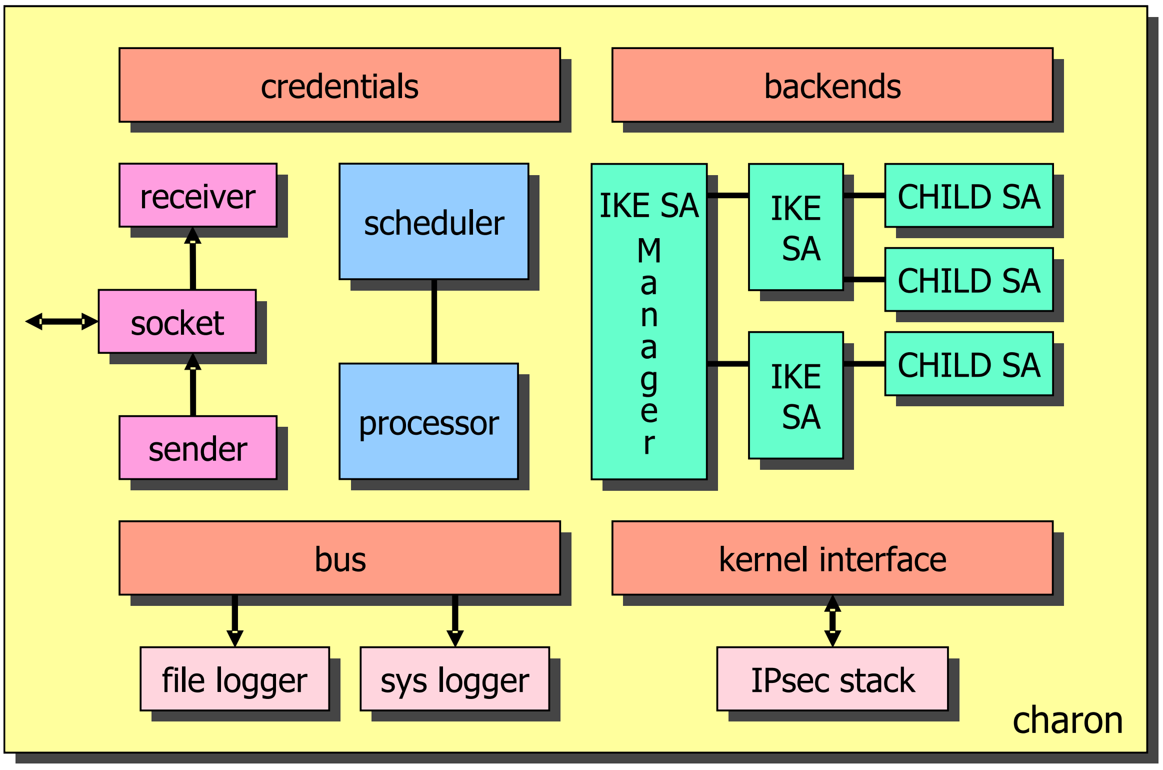


*Illustration 5: Software architecture of the IKEv2 charon daemon*

## Configuration and Control – the FreeS/WAN Way

```
#ipsec.secrets for roadwarrior carol

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

```
#ipsec.conf for roadwarrior carol

conn home
     keyexchange=ikev2
     authby=psk
     left=%defaultroute
     leftsourceip=%config
     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 "jVzONCF02ncsgiSlmIXeqhGN"
```

```
#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
     rightsourceip=10.3.0.0/16
     auto=add
```

*Illustration 6: IKEv2 Mixed PSK/RSA Authentication*

Illustration 6 shows a typical strongSwan connection definition using the classical configuration files *ipsec.conf* for connection and peer information and *ipsec.secrets* for credentials. Both file formats go a long way back to the original FreeS/WAN project and have been kept by the strongSwan project with only some extensions added. What is totally new, is the way the configuration files are used to set up an IKEv2 connection with a mixed PSK / RSA authentication plus the automatic assignment of virtual IP addresses for the road warrior clients from a common address pool. This goal is achieved with the following configuration parameters:

- **keyexchange=ikev2** This parameter has always existed but had been used in the past with its default value *"ike"* that stood for *"ikev1"*. With the setting *"ikev2"* the connection is set up by the IKEv2 *charon* daemon in place of the default IKEv1 *pluto* daemon.
- **authby=psk, authby=rsasig** This is an IKEv2 specialty where the road warriors *carol* and *dave* (see Illustration 2 for the network topology) use a Pre-Shared Key (PSK) each as user credentials, whereas the VPN gateway *moon* authenticates itself with a digital RSA signature corroborated by an X.509 host certificate.
- **leftsourceip=%config, rightsourceip=10.3.0.0/16** With the *leftsourceip=%config* setting the road warriors *carol* and *dave* request a virtual IP address from the central gateway *moon*, to by used as a source address within the IPsec tunnel. The gateway assigns such addresses in increasing order on a first-come, first-served basis out of a pool defined by the *rightsourceip* statement. If addresses are to be permanently bound to a given identity, the notation *rightsourceip=%<poolname>* is used which stores the lease information in a non-volatile database.

Going back to the block diagram in Illustration 5, we see that the *stroke* back end is used to implement the connection information retrieval and the fetching of credentials based on the *ipsec.conf* and *ipsec.secrets* files in a classical FreeS/WAN way. In addition to that the stroke protocol is employed for control and logging functions as shown in the example of Illustration 7. The *ipsec start* command not only starts the *ipsec starter* which in turn forks the *pluto* and *charon* daemons but due to the *auto=start* directive in *carol*'s *ipsec.conf* file automatically initiates the connection setup to gateway *moon* via a *stroke up* command. The resulting log entries show that the IKEv2 protocol requires only **four** messages to set up an IKE_SA including the first CHILD_SA which is far more efficient than the **nine** messages required by the old IKEv1 protocol.

```
carol> ipsec start

05[AUD] initiating IKE_SA 'home' to 192.168.0.1
05[ENC] generating IKE_SA_INIT request 0 [SA KE No 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 0 [SA KE No N N]
06[ENC] generating IKE_AUTH request 1 [IDi CERTREQ 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 1 [IDr CERT AUTH CP SA TSi TSr N]
07[ENC] IKE_SA 'home' established between 192.168.0.100...192.168.0.1
07[IKE] installing new virtual IP 10.3.0.1
07[AUD] CHILD_SA 'home' established successfully
```

*Illustration 7: Using the stroke interface to start a connection*

The second example depicted in Illustration 8 shows the use of the *ipsec status* command which is actually translated into and executed as *ipsec stroke status* and shows some important charon system parameters, the connection definitions and the actual status of the IKE and IPsec security associations.

```
carol> ipsec status

Performance:
  uptime: 5 seconds, since Apr 28 18:30:36 2008
  worker threads: 11 idle of 16, job queue load: 1, scheduled events: 5
Listening IP addresses:
  192.168.0.100
  fec0::10
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: 15993ec81138c1b1_i* ce054ec02da36c8e_r, reauth in 51 minutes
  home{1}: INSTALLED, TUNNEL, ESP SPIs: c51cf634_i cf2c3efd_o
  home{1}: AES_CBC-128/HMAC_SHA1_96, rekeying in 14 minutes, last use: 2s_i 2s_o
  home{1}: 10.3.0.1/32 === 10.1.0.0/16
```

*Illustration 8: Using the stroke interface to get status information*

## Configuration and Control – the Modular Way

The strongSwan 4.2 branch released in April 2008 introduced some fundamental changes in the software architecture of the IKEv2 *charon* daemon. The logical building blocks *controller*, *credentials*, *backends*, *bus*, and *eap* have been completely modularized as is shown in the block diagram of Illustration 9. Charon's external interfaces can now be adapted and expanded at run-time by dynamically loading customized plugins.

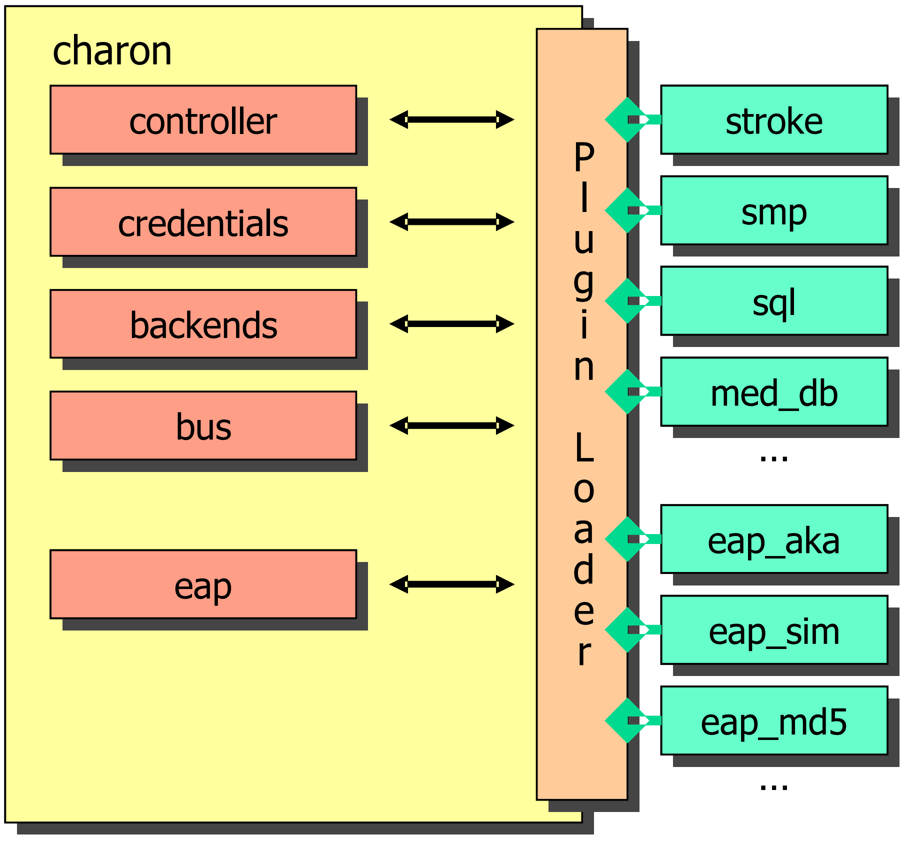


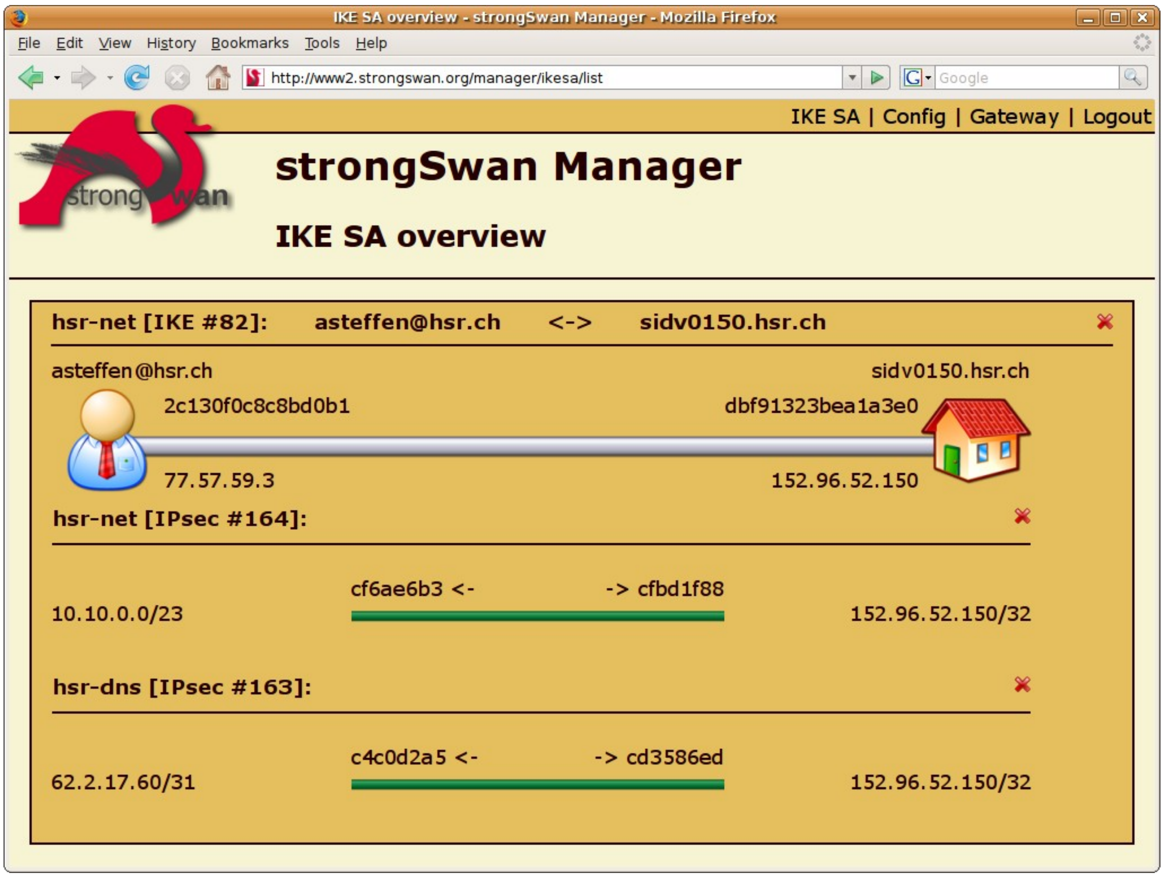
*Illustration 9: Modular structure of the charon daemon based on plugins*

Currently the following standard plugins are available:

- **stroke** is the only plugin loaded by default and implements the classical FreeS/WAN look and feel interface described in the previous section. The *stroke* plugin can be completely removed if not needed.
- **smp** implements the XML-based *strongSwan Management Protocol (SMP)* which uses a UNIX socket as a bi-directional communications channel. *strongSwan Manager*, our web-based demo application will be presented in the next section.
- **sql** implements an SQL database back end for storing configuration data and credentials. The corresponding entity-relationship diagrams will be presented later in the text.
- **med_db** implements the database interface of the *strongSwan Mediation Manager* presented later in this text.
- **eap_x** plugins implement any number of Extended Authentication Protocols (EAP). Available are EAP-AKA, EAP-SIM, and EAP-MD5.

## strongSwan Manager

The *strongSwan Manager* prototype web application depicted in Illustration 10 uses the XML-based *strongSwan Management Protocol (SMP)* to get current connection status information from the *charon* daemon. SMP uses *Relax-NG* schemata (www.relaxng.org) for validation, since they are more more powerful than DTD but simpler than full-blown XML schemata. The graphical web interface is completely coded in the C programming language but built with the help of the *ClearSilver* HTML template system (www.clearsilver.net) which tremendously facilitates the page design. The *strongSwan Manager* can be called e.g. by an *Apache* or *lighttpd* web server via its standard *FastCGI* interface (www.fastcgi.com).


IKE SA overview - strongSwan Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www2.strongswan.org/manager/ikesa/list
Google
IKE SA | Config | Gateway | Logout
strongSwan Manager
IKE SA overview

hsr-net [IKE #82]: asteffen@hsr.ch <-> sidv0150.hsr.ch

asteffen@hsr.ch sidv0150.hsr.ch
2c130f0c8c8bd0b1 dbf91323bea1a3e0
77.57.59.3 152.96.52.150

hsr-net [IPsec #164]:

10.10.0.0/23 cf6ae6b3 <- -> cfbd1f88 152.96.52.150/32

hsr-dns [IPsec #163]:

62.2.17.60/31 c4c0d2a5 <- -> cd3586ed 152.96.52.150/32

*Illustration 10: strongSwan manager displaying active IKE SAs*

The *strongSwan Manager* can also be used to start and stop individual IKE security associations and/or dependent CHILD SAs via the SMP interface, thus replacing the *ipsec up* and *ipsec down* stroke commands. In Illustration 10 a click on the top-most red cross symbol takes down the IKE SA #82 together with the two CHILD SAs #163 and #164, whereas a click on the lowest cross terminates the CHILD SA *hsr-dns* only.

Another feature of the strongSwan Manager is the enumeration of all available connection definitions by means of the *Config* menu. Individual connections can then be started from this menu by the press of a button. Configuration information is currently retrieved from the *charon* daemon via the SMP interface. Future version of the strongSwan Manager will allow to edit connections via an SQL database interface.

## SQL Database Interface

The modular structure of the *charon* daemon presented in Illustration 9 listed the possibility of retrieving connection information and credentials via a database back end implemented by means of the *sql* plugin. The actual physical database access and the SQL queries are realized by registering a product-specific driver with the *libstrongswan* library as shown in the plugin overview of Illustration 12. Currently only SQLite (www.sqlite.org) and MySQL (www.mysql.com) are supported but in principle any other relational database could be connected by writing an appropriate driver plugin.

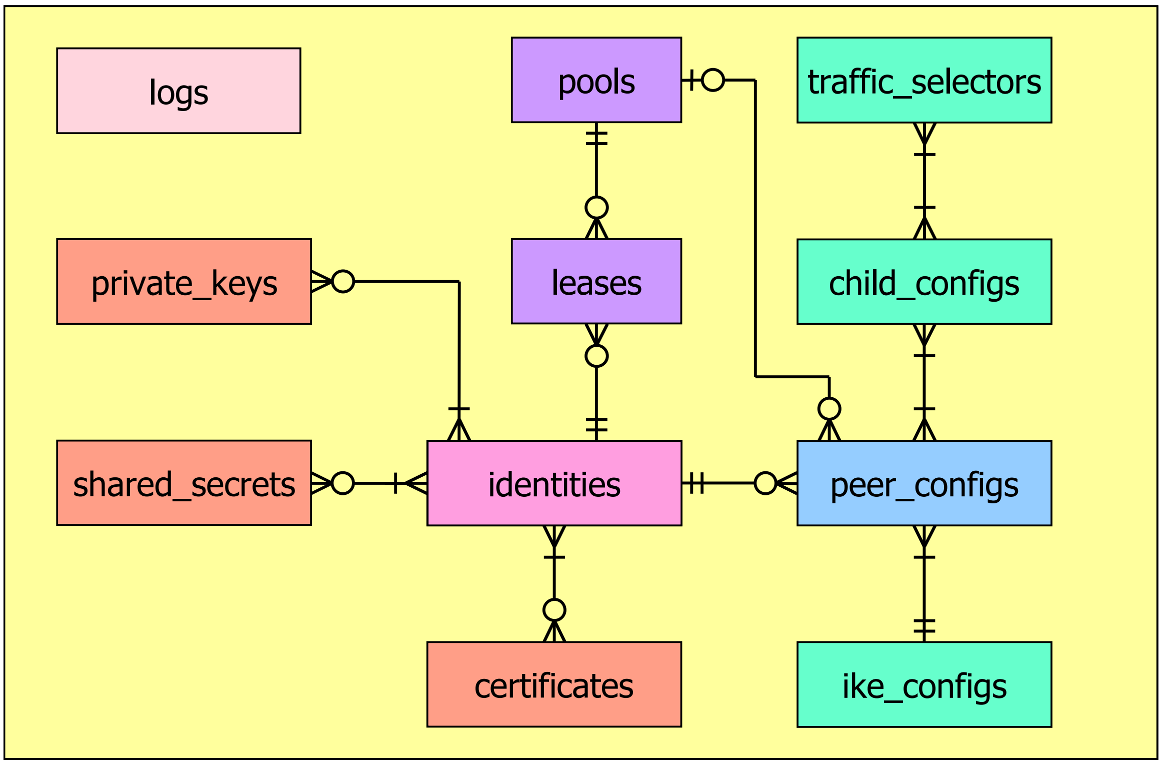


*Illustration 11: strongSwan entity relationship diagram*

The entity-relationship diagram used by both the SQLite and MySQL plugins is shown in Illustration 11. The diagram defines entities for storing

- **credentials** comprising tables for shared_secrets, private_keys, and certificates bound to one or several identities
- **identities** of various types (ID_FQDN, ID_RFC822_ADDR, ID_IPV4_ADDR, ID_IPV6_ADDR, ID_DER_ASN1_DN, ID_KEY_ID)
- **connections definitions** consisting of peer_configs, ike_configs, child_configs, and traffic_selectors
- **pools** of virtual IPv4 or IPv6 addresses that can be leased persistently to specific identities.

Currently no strongSwan-specific tool exists for entering and editing the information contained in the database although there are plans to build such administration capabilities into future releases of the *strongSwan Manager* web application.

## Plugins for the “libstrongswan” Library

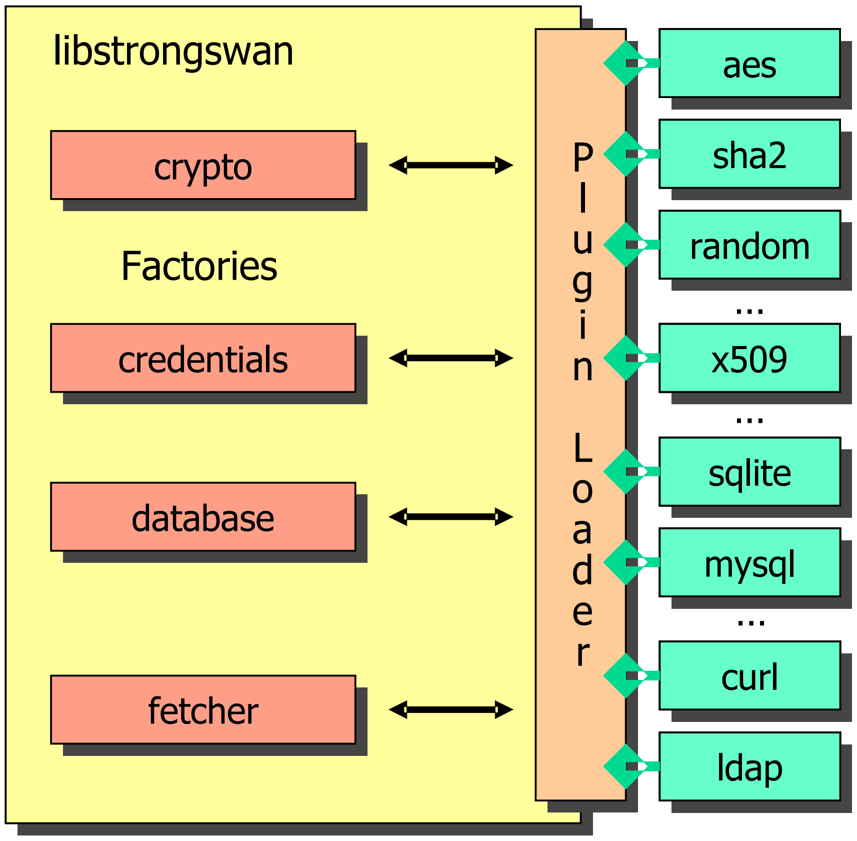


*Illustration 12: Modular structure of libstrongswan based on plugins*

The *libstrongswan* library has been modularized into factories and various plugins that register with them. The following modules are currently available:

- **aes** implements AES with 128/192/256 bit keys in SW (non-US code)
- **des** implements single and triple-DES in SW (non-US code)
- **md5** implements the MD5 hash function in SW
- **sha1** implements the SHA-1 hash function in SW
- **sha2** implements the SHA-256/384/512 hash function in SW
- **padlock** accelerates AES and SHA-1 in HW (uses VIA PadLock processor)
- **hmac** implements the HMAC authentication function in SW
- **xcbc** implements the AES XCBC MAC function in SW
- **fips_prf** implements the FIPS PRF function in SW
- **random** generates random numbers (uses /dev/random)
- **gmp** implements RSA and DH public key cryptography (uses libgmp)
- **openssl** implements cryptographic functions using OpenSSL
- **x509** implements support ofX.509 certificates, CRLs, and OCSP
- **mysql** wraps the access to MySQL databases (uses libmysqlclient_r)
- **sqlite** wraps the access to SQLite databases (uses libsqlite3)
- **curl** fetches data from HTTP and other URLs (uses libcurl)
- **ldap** fetches of data from LDAP servers (uses OpenLDAP)

## Cryptographic Plugins

As shown in Illustration 12 all cryptographic functions that were hard-coded in earlier versions of the *libstrongswan* library have been modularized and can now be individually replaced by other implementations. Currently there are two projects under way that intend to offer alternative cryptographic plugins:

- Use of the OpenSSL FIPS 140-2 library (http://www.openssl.org/docs/fips/)

  Especially in the United States there is a big interest in a FIPS 140-2 certified strongSwan variant. The easiest way to achieve this goal is to use the FIPS 140-2 certified openssl-fips-1.1.2 library for all IKEv2 cryptographic operations.

- Use of cryptographic hardware accelerators

  On VPN gateways with thousands of concurrent connections there might occur the need to accelerate the most time-consuming cryptographic operations of the IKEv2 protocol that take place in the userland (in contrast to the cryptographic operations on ESP packets that are handled by the Crypto API in the Linux 2.6 kernel). As a proof of concept the strongSwan 4.2 distribution comes with a padlock plugin which delegates AES encryption of IKE packets and SHA-1 and SHA-256 hashes to the PadLock Security Engine which is available on the VIA EPIA Mini- and Nano-ITX boards (see Illustration 13). Actually the speedup due to AES and SHA acceleration is rather negligible. Much greater savings could be expected by using the hardware Montgomery Multiplier for RSA and Diffie-Hellman public key operations.

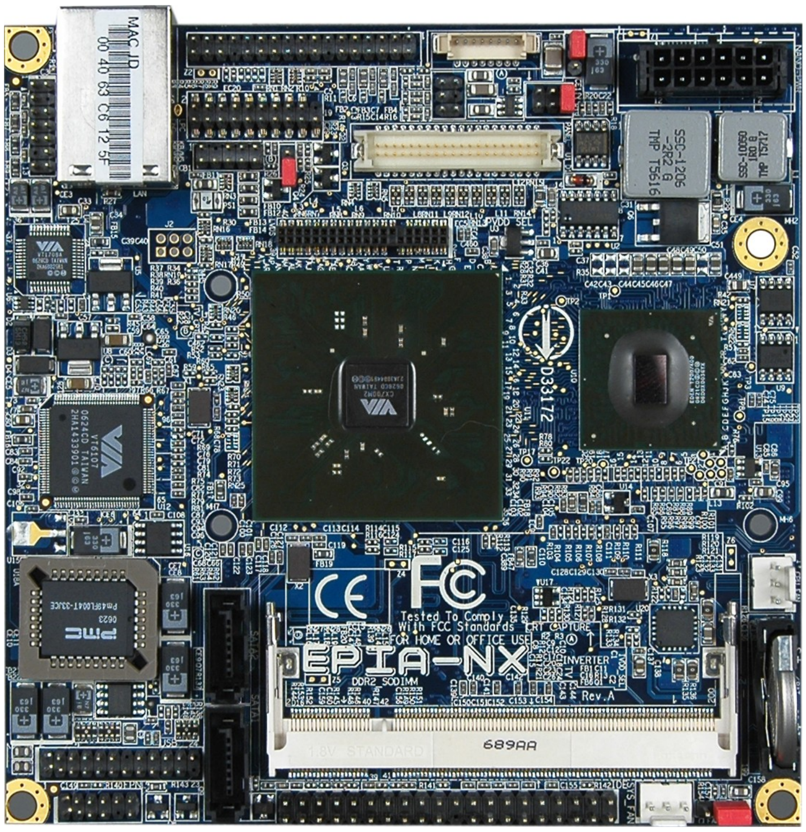


*Illustration 13:  VIA EPIA-NX Nano-ITX board with PadLock Security Engine*

## The IKEv2 Mediation Service

In December 2006 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 *<draft-brunner-ikev2-mediation-00.txt>* that was published by the IETF on April 16 2008 (see Illustration 15).

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. In our scheme this mediation server will be using the IKEv2 protocol instead, as shown in Illustration 14.

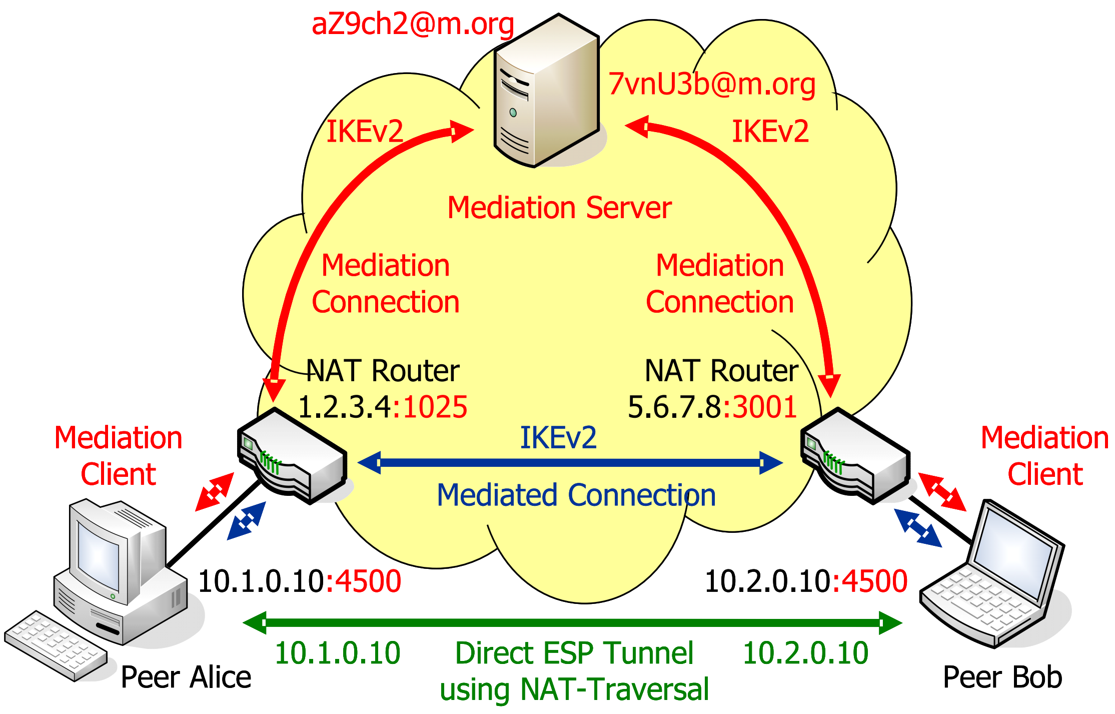


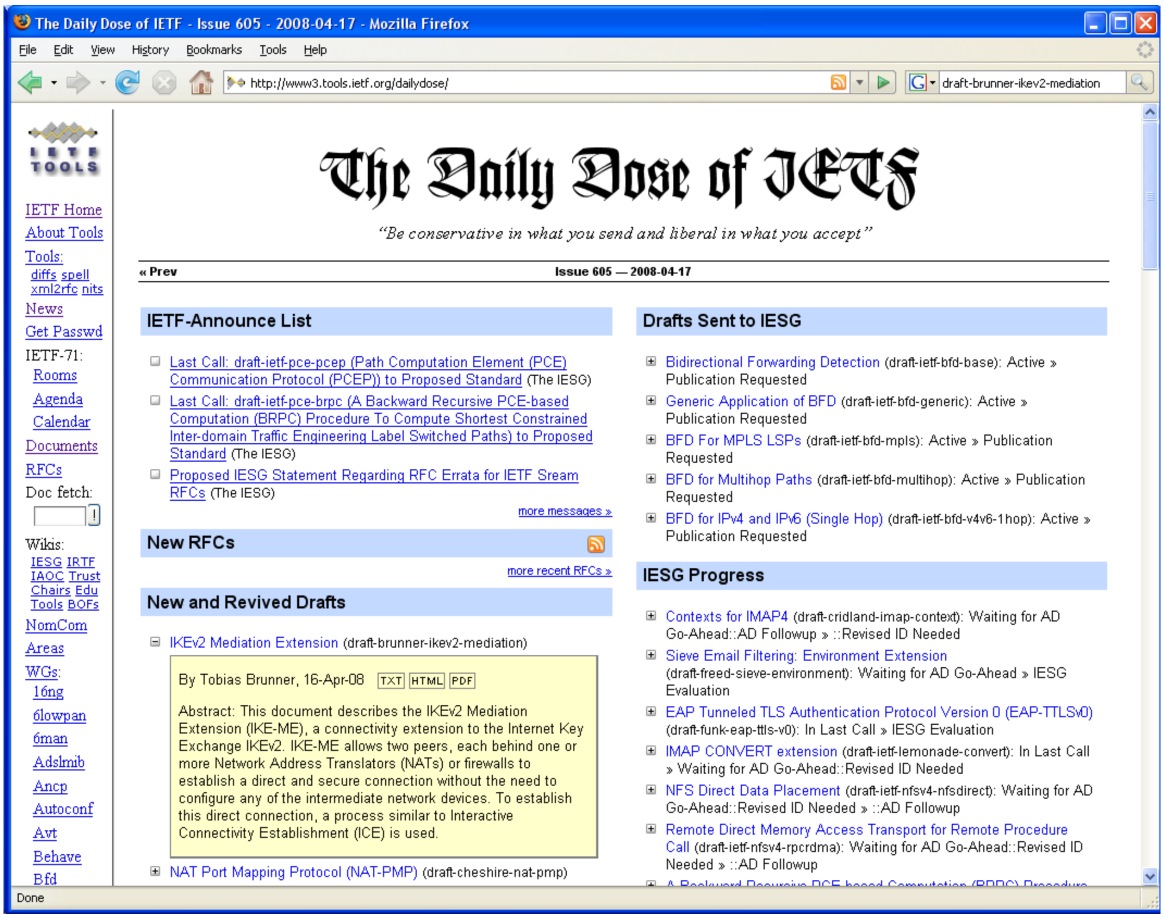
*Illustration 14: Peer-to-peer NAT-traversal for IPsec*

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. *alice* 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.



The Daily Dose of IETF

"Be conservative in what you send and liberal in what you accept"

« Prev — Issue 605 — 2008-04-17

**IETF-Announce List**

- Last Call: draft-ietf-pce-pcep (Path Computation Element (PCE) Communication Protocol (PCEP)) to Proposed Standard (The IESG)
- Last Call: draft-ietf-pce-brpc (A Backward Recursive PCE-based Computation (BRPC) Procedure To Compute Shortest Constrained Inter-domain Traffic Engineering Label Switched Paths) to Proposed Standard (The IESG)
- Proposed IESG Statement Regarding RFC Errata for IETF Sream RFCs (The IESG)

more messages »

**New RFCs**

more recent RFCs »

**New and Revived Drafts**

- IKEv2 Mediation Extension (draft-brunner-ikev2-mediation)

  By Tobias Brunner, 16-Apr-08 TXT HTML PDF

  Abstract: This document describes the IKEv2 Mediation Extension (IKE-ME), a connectivity extension to the Internet Key Exchange IKEv2. IKE-ME allows two peers, each behind one or more Network Address Translators (NATs) or firewalls to establish a direct and secure connection without the need to configure any of the intermediate network devices. To establish this direct connection, a process similar to Interactive Connectivity Establishment (ICE) is used.
- NAT Port Mapping Protocol (NAT-PMP) (draft-cheshire-nat-pmp)

**Drafts Sent to IESG**

- Bidirectional Forwarding Detection (draft-ietf-bfd-base): Active » Publication Requested
- Generic Application of BFD (draft-ietf-bfd-generic): Active » Publication Requested
- BFD For MPLS LSPs (draft-ietf-bfd-mpls): Active » Publication Requested
- BFD for Multihop Paths (draft-ietf-bfd-multihop): Active » Publication Requested
- BFD for IPv4 and IPv6 (Single Hop) (draft-ietf-bfd-v4v6-1hop): Active » Publication Requested

**IESG Progress**

- Contexts for IMAP4 (draft-cridland-imap-context): Waiting for AD Go-Ahead::AD Followup » ::Revised ID Needed
- Sieve Email Filtering: Environment Extension (draft-freed-sieve-environment): Waiting for AD Go-Ahead » IESG Evaluation
- EAP Tunneled TLS Authentication Protocol Version 0 (EAP-TTLSv0) (draft-funk-eap-ttls-v0): In Last Call » IESG Evaluation
- IMAP CONVERT extension (draft-ietf-lemonade-convert): In Last Call » Waiting for AD Go-Ahead::Revised ID Needed
- NFS Direct Data Placement (draft-ietf-nfsv4-nfsdirect): Waiting for AD Go-Ahead::Revised ID Needed » ::AD Followup
- Remote Direct Memory Access Transport for Remote Procedure Call (draft-ietf-nfsv4-rpcrdma): Waiting for AD Go-Ahead::Revised ID Needed » ::AD Followup

*Illustration 15: draft-brunner-ikev2-mediation released on April 16 2008*

Tobias Brunner integrated his *IKEv2 Mediation Extension* into the current strongSwan 4.2 distribution so that it is publicly available for everyone to experiment with. A sample scenario can be found under the link

http://www.strongswan.org/uml/testresults42/p2pnat/medsrv-psk/

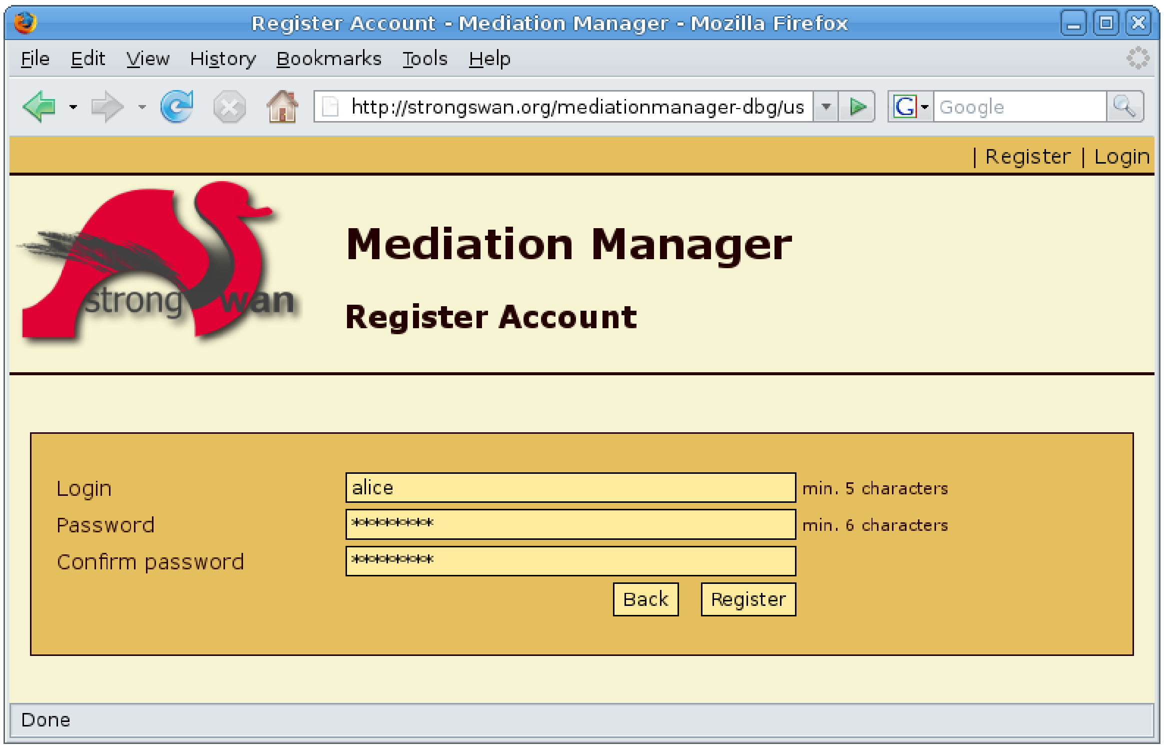
Feedback from visitors at the strongSwan booth at last year's LinuxTag has shown though, that manual configuration of the *mediation* and *mediated connections* via *ipsec.conf* and *ipsec.secrets* are too complex for the average user. Therefore we decided to develop a user-friendly graphical tool, the *strongSwan Mediation Manager* which we are going to present in the next section. The actual implementation is currently being done by the HSR students Philip Bötschi and Adrian Dörig as part of their bachelor thesis.

## The strongSwan Mediation Manager

The *strongSwan Mediation Manager* uses the same FastCGI web technology as the *strongSwan Manager* shown in Illustration 10. Actually there are two slightly different web applications:

- The **strongSwan Mediation Server Manager** which allows a user to register her client[s] that are hidden behind a NAT router with the strongSwan Mediation server.
- The **strongSwan Mediation Client Manager** which allows a user to configure both the mediated peer-to-peer connection as well as the mediation connection to the mediation server on a strongSwan client.

The prerequisites for both the server and client applications are a HTTP server (e.g. lighttpd) and a database (SQLite or MySQL) running on the target system.


Register Account - Mediation Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://strongswan.org/mediationmanager-dbg/us
Google
| Register | Login
strongSwan
Back
Register
Done

Mediation Manager

Register Account

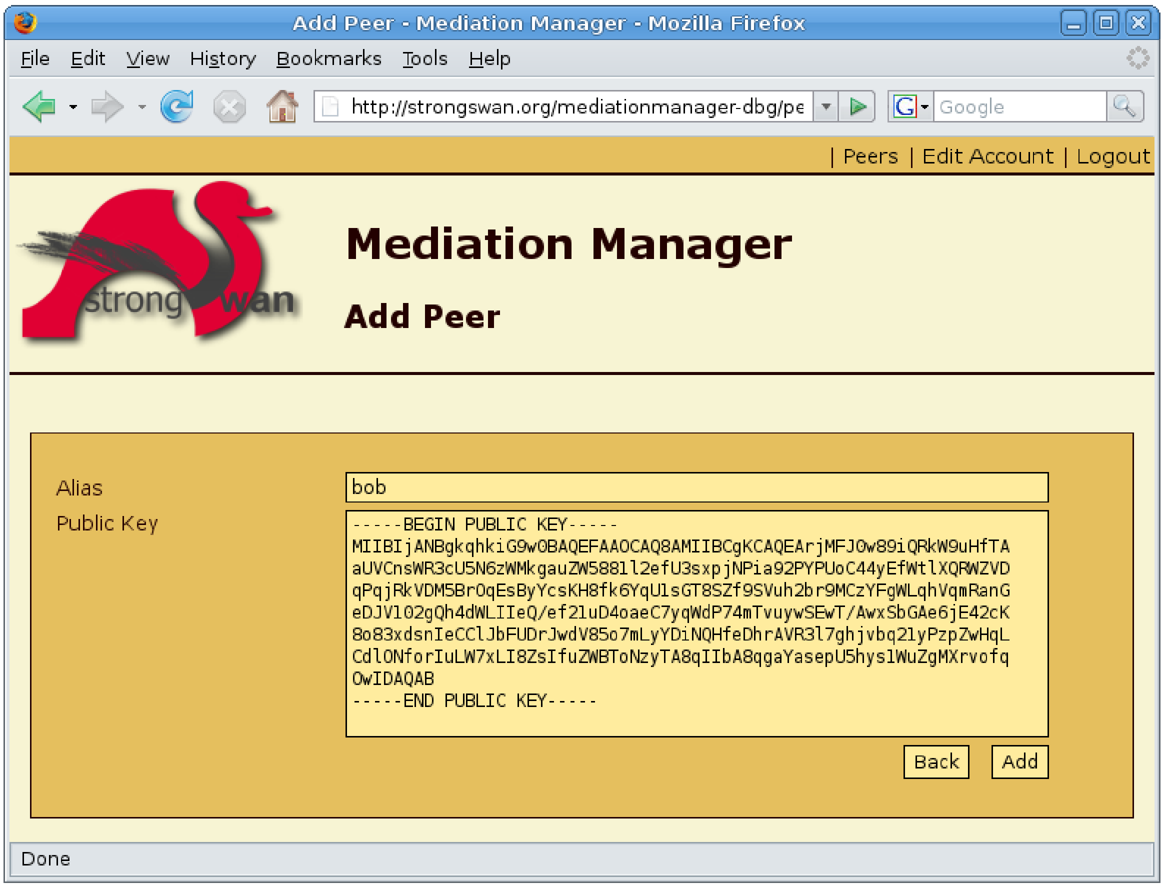
| | | |
|---|---|---|
| Login | alice | min. 5 characters |
| Password | ********* | min. 6 characters |
| Confirm password | ********* | |

*Illustration 16: Login at the strongSwan mediation manager*

Illustration 16 shows the registration screen of the *strongSwan Mediation Server Manager*. Users that want to register one or more clients with the mediation server must open a personal user account first. Currently the user identity is not verified so that anyone can use the mediation service.

After a successful login the *Add Peer* screen depicted in Illustration 17 allows a client to be registered with the mediation server. In order to keep the complexity for the average user low, peer authentication is based on raw 2048 bit RSA key pairs that are automatically generated by the *Mediation Client Manager* and can be transferred via simple copy-and-paste into the *Mediation Server Manager*'s *Add Peer* menu. The RSA public key format is identical to that used by OpenSSL (www.openssl.org). The peer identity is an ID_KEY_ID derived from the peer's public key which guarantees a high degree of anonymity and eliminates the risk of peer identity collisions.

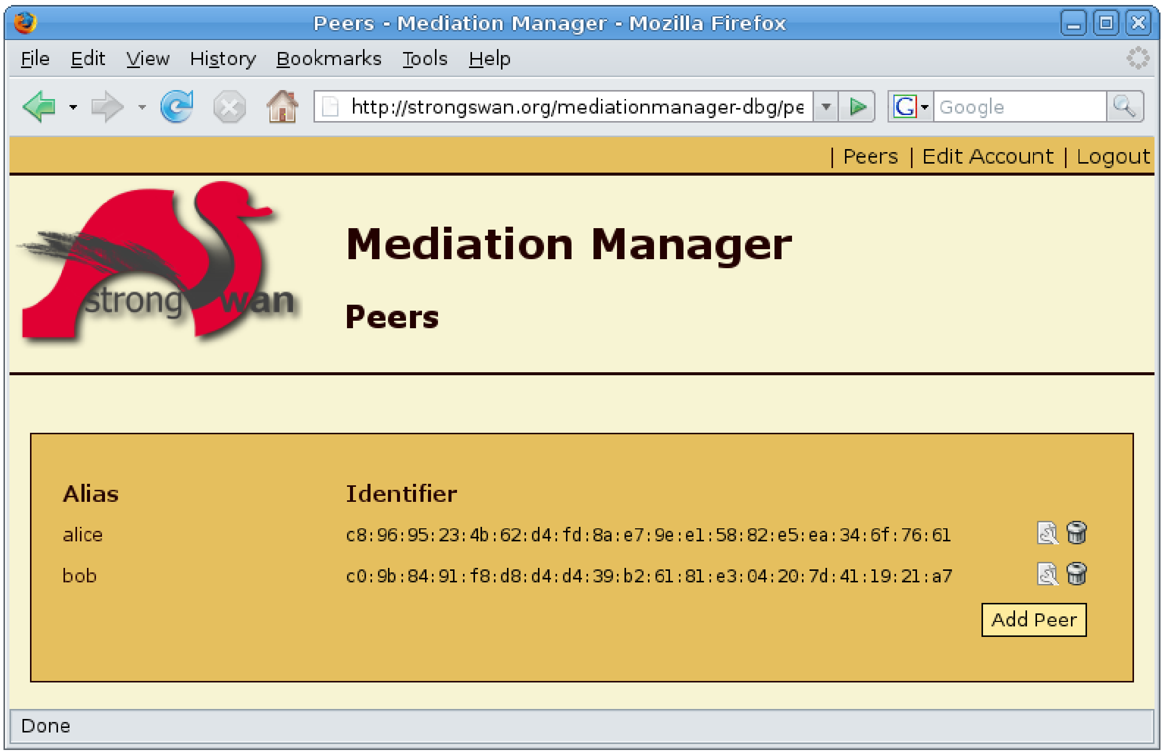
Two examples of peer identities are shown in the *Peers* menu of Illustration 18.


Add Peer - Mediation Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://strongswan.org/mediationmanager-dbg/pe
Google
| Peers | Edit Account | Logout
strongswan
Mediation Manager
Add Peer
Back
Add
Done

| | |
|---|---|
| Alias | bob |
| Public Key | |

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEArjMFJ0w89iQRkW9uHfTA
aUVCnsWR3cU5N6zWMkgauZW588ll2efU3sxpjNPia92PYPUoC44yEfWtlXQRWZVD
qPqjRkVDM5Br0qEsByYcsKH8fk6YqUlsGT8SZf9SVuh2br9MCzYFgWLqhVqmRanG
eDJVl02gQh4dWLIIeQ/ef2luD4oaeC7yqWdP74mTvuywSEwT/AwxSbGAe6jE42cK
8o83xdsnIeCClJbFUDrJwdV85o7mLyYDiNQHfeDhrAVR3l7ghjvbq2lyPzpZwHqL
CdlONforIuLW7xLI8ZsIfuZWBToNzyTA8qIIbA8qgaYasepU5hys1WuZgMXrvofq
OwIDAQAB
-----END PUBLIC KEY-----
```

*Illustration 17: Register a peer with the mediation manager*


Peers - Mediation Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://strongswan.org/mediationmanager-dbg/pe
Google
| Peers | Edit Account | Logout
strongswan
Mediation Manager
Peers
Add Peer
Done

| Alias | Identifier |
|---|---|
| alice | c8:96:95:23:4b:62:d4:fd:8a:e7:9e:e1:58:82:e5:ea:34:6f:76:61 |
| bob | c0:9b:84:91:f8:d8:d4:d4:39:b2:61:81:e3:04:20:7d:41:19:21:a7 |

*Illustration 18: List of registered peers*

## Conclusion

The modularity and scalability offered by the new strongSwan 4.2 branch is hopefully going to form the basis for many industry-grade VPN applications. The flexible concept of library plugins opens up a broad choice of cryptographic hardware and software options. SQL-based database interfaces and first attempts at a graphical management interface are going to facilitate the administration of VPN networks. And finally strongSwan's unique approach to solve the NAT traversal problems by means of the IKEv2 Mediation Service is expected to find manifold applications in remote control and diagnosis systems.

## Bibliography

- IETF RFC 2409 „*Internet Key Exchange (IKE)”*, 1998
- IETF RFC 4306 „*Internet Key Exchange (IKEv2) Protocol*“, 2005
- IETF Draft <draft-brunner-ikev2-mediation-00.txt> „*IKEv2 Mediation Extension*“, 2008