Quick Note 62
IKEv2 IPsec VPN from TransPort WR to StrongSwan using Certificates
1  INTRODUCTION

1.1  Outline

This document describes how to configure a VPN IPsec tunnel between a Digi TransPort WR to a StrongSwan server IKEv2 using Certificates authentication.

The network diagram considered in this example is the following:

This guide details the steps involved in configuring a Digi TransPort router to act as an IPsec VPN client to a StrongSwan appliance configured as an IPsec VPN server using IKEv2 and Certificates authentication.

1.2  Assumptions

This guide has been written for technically competent personnel who are able to configure a standard IPSec tunnel between 2 TransPort WR routers and are familiar with the use of routing protocols.

This application note applies to:

Model: Digi TransPort WR11/21/31/41/44

Firmware versions:

WR21: 5.2.17.10 and later

Configuration: This document assumes that the devices are set to their factory default configurations. Most configuration commands are shown only if they differ from the factory default.
1.3 Corrections

Requests for corrections or amendments to this application note are welcome and should be addressed to: tech.support@digi.com.

Requests for new application notes can be sent to the same address.

1.4 Version

<table>
<thead>
<tr>
<th>Version Number</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Published</td>
</tr>
</tbody>
</table>
2 STRONGSWAN INSTALLATION AND CERTIFICATES CREATION

2.1 Installing StrongSwan

The following instructions refer to an Ubuntu system for the server side, please refer to [http://www.strongswan.org](http://www.strongswan.org) for installation on other Linux distributions.

From the terminal, issue the command "sudo apt-get install strongswan":

```bash
digi@Digi:~$ sudo apt-get install strongswan
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  accountsservice-ubuntu-schemas accountsservice-ubuntu-touch-schemas address-book-service
  biometrydb-bin evolution-data-server-utouch folks-common history-service indicator-transfer
  indicator-transfer-download-manager libbiometryd1 libboost-system1.61.0 libcgmanager libclick-0.4-0
  libconnexion-qt1 libdbus-cpp5 libfolks-eds25 libfolks25 libhistoryservice libindicator-transfer
  libindicator-transfer0 libbleveldev1v5 liblightdm-tqt5-3-0 libmediascanner-2.0-4 libmiral2
  libmission-control-plugins0 libonline-accounts-daemon1 libonline-accounts-qt5 libopkpoppler-qt5-1
  libpqjdbc-dbg
  libgspell-common libgspell3 libgstreamer1.0-ubuntu4.4 libgstreamer-plugins-base1.0-ubuntu4.4
  libgstreamer-plugins-good1.0-ubuntu4.4 libgstreamer-plugins-ugly1.0-ubuntu4.4 libgstreamer-plugins-ugly1v5
  libgstreamer-plugins-videoproc1.0-ubuntu4.4 libgstreamer-plugins-video1.0-ubuntu4.4 libgstreamer-plugins-good1.0-
  gstreamer-plugins-ugly1.0-ubuntu4.4 libgstreamer-plugins-ugly1v5
  libgnome-charactermap4 libgnome-control-center1 libgnome-crew1 libhistoryservice-qt5 libindicator
  libindicator-0.4-0 liblightdm-tqt5-3-0 libmediascanner-2.0-4 libmiral2
  libmission-control-plugins0 libonline-accounts-daemon1 libonline-accounts-qt5 libopkpoppler-qt5-1
  libpqjdbc-dbg
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  strongswan
0 upgraded, 1 newly installed, 0 to remove and 115 not upgraded.
Need to get 26,8 kB of archives.
After this operation, 175 kB of additional disk space will be used.
Get:1 http://de.archive.ubuntu.com/ubuntu zesty-updates/main amd64 strongswan all 5.5.1-
1ubuntu3.2 [26,8 kB]
Fetched 26,8 kB in 0s (80,2 kB/s)
Preparing to unpack .../strongswan_5.5.1-1ubuntu3.2_all.deb ...
Unpacking strongswan (5.5.1-1ubuntu3.2) ...
NOTE: The StrongSwan software is now installed and ready to use. In this document, the IPSec VPN will use Certificates authentication. If the set of certificates/keys is already available, you can skip until section 3.
```
### 2.2 Installing the PKI tool

In this example the StrongSwan PKI tool will be used to generate the Certificates and Keys needed for the authentication of the tunnel. This tool can be installed using the command “sudo apt install strongswan-pki”:

```bash
digi@Digi:~$ sudo apt install strongswan-pki
```

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  accountsservice-ubuntu-schemas accountsservice-ubuntu-touch-schemas address-book-service
  biometryd-bin evolution-data-server-utouch folks-common history-service indicator-transfer
  indicator-transfer-download-manager libbiometryd1 libboost-system1.61.0 libcmanager0 libclic
  k0.4-0 libconnectivity-qt1 libdbus-cpp5 libfolks-eds25 libfolks25 libhistoryservice0
  libindicator-transfer0 libleveldb1v5 liblightdm-qt5-3-0 libmediascanner-2.0-4 libmiral2
  libmission-control-plugins0 libonline-accounts-daemon1 libonline-accounts-q1 libpoppler-qt5-1
  libpqjdbc2-d8
  libsqlitedb-1-0 libsqlitedb1v5 libsystemsettings1 libtelepathy-qt4-2 libtelepathy-qt5-0 libthumbn
 ailer-q1 libtrust-store2 libubuntu-location-service3 libusermetricsinput1 libusermetricsoutput1 lib
  xheaders4.10.0-19 generic linux-header4.10.0-19-generic linux-header4.10.0-32 generic linux-hea
  ders4.10.0-32 generic linux-header4.10.0-32-generic lin
  xheaders4.10.0-19-generic linux-header4.10.0-32-generic lin
  xheader4.10.0-32 generic mediascanner2.0 mir-client-platform-mesa5 mir-graphic
  s-drivers-desktop mir-platform-graphics-mesa-kms12 mir-platform-graphics-mesa-x12 mir-platform
  input-evdev6
  policykit-unity8 qmenu-model qml qml-module-biometryd qml-module-ofono qml-module-pa
 uthentication0.1 qml-module-qmltermwidget1.0 qml-module-qtmultimedia qml-module-qtmultimed
  a42
  qml-module-qtmultimediawidgets5 qml-module-qtmultimedia qml-module-qtqml-state
  machine qml-module-qtmultimedia qml-module-ubunto-connectivity qml-module-ubuntu-o
  nlineaccounts2 qml-module-ubuntu-settings-components qml-module-ubuntu-thumbnailer0.1 qtmult
  iuml widget5 galera
  qtdeclarative5-gsettings1.0 qtdeclarative5-qtmir-plugin qtdeclarative5-ubuntu-settings-co
  mponents qtdeclarative5-ubuntu-telephony0.1 qtdeclarative5-unity-notifications-plugin qtmir
  desktop
  qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
  rial qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnameterminal qtnamet
2.3 Create a Root CA Certificate

First the CA key is created:

```
digi@Digi:~$ ipsec pki --gen --outform pem > caKey.pem
```

Where “caKey.pem” will be the name of the key that will be created.

With that, the CA certificates can be created:

```
digi@Digi:~$ ipsec pki --self --in caKey.pem --dn "C=DE, O=Digi, CN=digiCA" --ca --outform pem > caCert.pem
```

Where:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>caKey.pem</td>
<td>CA Key just created</td>
</tr>
<tr>
<td>C=DE</td>
<td>The two-letter ISO 3166 abbreviation for your country.</td>
</tr>
<tr>
<td>O=Digi</td>
<td>The exact legal name of your organization. Do not abbreviate your organization name.</td>
</tr>
<tr>
<td>CN=digiCA</td>
<td>Common Name for the CA</td>
</tr>
<tr>
<td>caCert.pem</td>
<td>Name of the CA certificate that will be created</td>
</tr>
</tbody>
</table>

The CA certificate just created, has to be moved to the `/etc/ipsec.d/cacerts` folder, in order to be correctly used by StrongSwan:

```
digi@Digi:~$ sudo cp caCert.pem /etc/ipsec.d/cacerts
```


### 2.4 Create a CA-Signed Server Certificate

The private Key for the Server is created with the following command:

```
digi@Digi:~$ ipsec pki --gen --outform pem > SrvKey.pem
```

The Server key just created, has to be moved to the `/etc/ipsec.d/private` folder, in order to be correctly used by StrongSwan:

```
digi@Digi:~$ sudo cp SrvKey.pem /etc/ipsec.d/private
```

Then, the Server Certificate can be created:

```
digi@Digi:~$ ipsec pki --pub --in SrvKey.pem | ipsec pki --issue --cacert caCert.pem --cakey caKey.pem --dn "C=DE,O=Digi, CN=server" --flag serverAuth --flag ikeIntermediate --san server --outform pem > SrvCert.pem
```

Where:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SrvKey.pem</td>
<td>Server Key just created</td>
</tr>
<tr>
<td>caCert.pem</td>
<td>CA certificate created in previous step</td>
</tr>
<tr>
<td>caKey.pem</td>
<td>CA key created in previous step</td>
</tr>
<tr>
<td>C=DE</td>
<td>The two-letter ISO 3166 abbreviation for your country.</td>
</tr>
<tr>
<td>O=Digi</td>
<td>The exact legal name of your organization. Do not abbreviate your organization name.</td>
</tr>
<tr>
<td>CN=server</td>
<td>Common Name for the Server</td>
</tr>
<tr>
<td>SrvCert.pem</td>
<td>Name of the Server certificate that will be created</td>
</tr>
</tbody>
</table>

The Server Certificate just created, has to be moved to the `/etc/ipsec.d/certs` folder, in order to be correctly used by StrongSwan:

```
digi@Digi:~$ sudo cp SrvCert.pem /etc/ipsec.d/certs
```
2.5 Create a CA-Signed Client Certificate

The private Key for the Client is created with the following command:

digi@Digi:~$ ipsec pki --gen --outform pem > CliKey.pem

Then, the Client Certificate can be created:

digi@Digi:~$ ipsec pki --pub --in CliKey.pem | ipsec pki --issue --cacert caCert.pem --cakey caKey.pem --dn "C=DE,O=Digi, CN=client" --san client --outform pem > ClientCert.pem

Where:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CliKey.pem</td>
<td>Server Key just created</td>
</tr>
<tr>
<td>caCert.pem</td>
<td>CA certificate created in previous step</td>
</tr>
<tr>
<td>caKey.pem</td>
<td>CA key created in previous step</td>
</tr>
<tr>
<td>C=DE</td>
<td>The two-letter ISO 3166 abbreviation for your country.</td>
</tr>
<tr>
<td>O=Digi</td>
<td>The exact legal name of your organization. Do not abbreviate your organization name.</td>
</tr>
<tr>
<td>CN=client</td>
<td>Common Name for the Server</td>
</tr>
<tr>
<td>ClientCert.pem</td>
<td>Name of the Server certificate that will be created</td>
</tr>
</tbody>
</table>
3 STRONGSWAN CONFIGURATION

3.1 LAN Interface

Configure the Local interface for the StrongSwan Server. In this example, the Ethernet interface used for LAN is called enp0s8:

```bash
digi@Digi:~$ sudo ifconfig enp0s8 10.0.1.100

digi@Digi:~$ sudo ifconfig enp0s8 netmask 255.255.255.0
```

3.2 WAN Interface

The ETH Interface used for WAN is named enp0s3 and is configured as a DHCP Client.

In order to check which IP is assigned the command “ifconfig” can be used:

```bash
digi@Digi:~$ ifconfig enp0s3

enp0s3:   flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
       inet 10.104.1.125  netmask 255.255.255.0  broadcast 10.104.1.255
       inet6 fe80::588c:91d1:d0f7:a888  prefixlen 64 scopeid 0x20<link>
       ether 08:00:27:e7:99:d4  txqueuelen 1000  (Ethernet)
       RX packets 287564  bytes 28927717 (28.9 MB)
       RX errors 0  dropped 0  overruns 0  frame 0
       TX packets 10841  bytes 1086181 (1.0 MB)
       TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

3.3 Uploading Certificates and Keys

Once you have the set of Certificates and Keys, that must be transferred to the system where the StrongSwan VPN server runs.

If the certificates have been created following section 2, all the files are already in the correct folders.

If not, be sure that the files are uploaded on the server as following:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Folder</th>
</tr>
</thead>
<tbody>
<tr>
<td>caCert.pem</td>
<td>CA certificate</td>
<td>/etc/ipsec.d/cacerts</td>
</tr>
<tr>
<td>SrvKey.pem</td>
<td>Server Key</td>
<td>/etc/ipsec.d/private</td>
</tr>
<tr>
<td>SrvCert.pem</td>
<td>Server Certificate</td>
<td>/etc/ipsec.d/certs</td>
</tr>
</tbody>
</table>
3.4 IKEv2/IPsec Tunnel Configuration

The IPsec configuration used by StrongSwan, is contained in two main configuration files:
- **ipsec.conf**: Used for Phase 1 (IKE) and Phase 2 IPsec configuration
- **ipsec.secrets**: Used for many types of secrets. In our example, it defines the Server Private Key to use.

### 3.4.1 ipsec.conf

Edit the ipsec.conf file using a text editor (nano is used in this example):

```plaintext
# ipsec.conf - strongSwan IPsec configuration file
# basic configuration
config setup
# Add connections here.

conn %default
  keyexchange=ikev2
  ike=aes256-sha1-modp1024
  esp=aes256-sha1
  dpdaction=clear
  dpddelay=300s

conn DigiWR
  left=%any
  leftcert=SrvCert.pem
  leftsubnet=10.0.1.0/24
  right=%any
  rightsubnet=10.0.0.0/24
  auto=add
```

Where:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conn %default</td>
<td></td>
<td>This tells Strongswan to use Ikev2. This parameter is used by default in strongswan 5.x</td>
</tr>
<tr>
<td>keyexchange</td>
<td>Ikev2</td>
<td></td>
</tr>
<tr>
<td>Ike</td>
<td>aes256-sha1-modp1024</td>
<td>This tells Strongswan to propose aes256 for encryption, sha1 for hashing, and DH group 2 for IKE.</td>
</tr>
<tr>
<td>esp</td>
<td>aes256-sha1</td>
<td>This tells Strongswan to propose aes256 for encryption and sha1 for hashing</td>
</tr>
<tr>
<td>dpdaction</td>
<td>clear</td>
<td>This means that when a Dead Peer is detected, the VPN will be closed</td>
</tr>
<tr>
<td>dpddelay</td>
<td>300s</td>
<td>Interval for DPD packets (seconds)</td>
</tr>
</tbody>
</table>

### 3.4.2 ipsec.secrets

Edit the ipsec.secrets file using a text editor (nano is used in this example):

```
# This file holds shared secrets or RSA private keys for authentication.
# RSA private key for this host, authenticating it to any other host
# which knows the public part.
: RSA SrvKey.pem
```

In this file, the private key to use for Server needs to be defined.

Please note that syntax is very important here, be sure to have the “:” with a **space** before the “RSA SrvKey.pem” part, otherwise this will not be recognized by StrongSwan.

### 3.4.3 Start the StrongSwan IPsec daemon

In order to have the changes to take effect, the StrongSwan IPsec daemon needs to be restarted using the following command:

```
digi@digi:~$ sudo ipsec restart
[sudo] password for digi:
Stopping strongSwan IPSec...
Starting strongSwan 5.5.1 IPSec [starter]...
```

---

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>left</strong></td>
<td>%any</td>
<td>That means that Strongswan accepts IKE connections on any local interfaces using any of it's locally configured IP's</td>
</tr>
<tr>
<td><strong>leftcert</strong></td>
<td>SrvCert.pem</td>
<td>Defines the Certificates file to use to authenticate the server itself</td>
</tr>
<tr>
<td><strong>leftsubnet</strong></td>
<td>10.0.1.0/24</td>
<td>Server subnet LAN</td>
</tr>
<tr>
<td><strong>right</strong></td>
<td>%any</td>
<td>That means that Strongswan accepts IKE connections coming from any IP address</td>
</tr>
<tr>
<td><strong>rightsubnet</strong></td>
<td>10.0.0.0/24</td>
<td>Client subnet LAN</td>
</tr>
<tr>
<td><strong>Auto</strong></td>
<td>add</td>
<td>That means that, when Strongswan first starts, it should accept this connection, but not try to initiate it.</td>
</tr>
</tbody>
</table>
4 TRANSPORT WR CONFIGURATION

4.1 LAN Interface
In this example, the LAN interface is configured with a static address as follows:

```
CONFIGURATION - NETWORK > INTERFACES > ETHERNET > ETH 0
```

![Configuration interface screenshot]

Where:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the following settings</td>
<td>Checked</td>
<td>A static IP will be used as defined below</td>
</tr>
<tr>
<td>IP Address</td>
<td>10.0.0.1</td>
<td>Enter the IP address of the LAN interface for the router</td>
</tr>
<tr>
<td>Mask</td>
<td>255.255.255.0</td>
<td>Enter the subnet mask</td>
</tr>
</tbody>
</table>
4.2 WAN Interface

In this example the WR Router has the Mobile interface as the WAN interface and it is configured as follows:

**CONFIGURATION - NETWORK > INTERFACES > MOBILE**

![Configuration Interface](image)

Where:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Plan/APN</td>
<td>internet.t-d1.de</td>
<td>Enter the APN of your mobile provider</td>
</tr>
</tbody>
</table>

*Please note:* Depending on provider, a SIM PIN or Username/Password may be required. If needed, enter them in the appropriate fields.
4.3 Uploading Certificates and Keys

Before to transfer the Certificates and Key files on the client, they must be renamed as follows:

<table>
<thead>
<tr>
<th>Filename</th>
<th>Purpose</th>
<th>New FileName</th>
</tr>
</thead>
<tbody>
<tr>
<td>caCert.pem</td>
<td>Root CA certificate</td>
<td>caCert.pem (*unchanged in this case)</td>
</tr>
<tr>
<td>ClientCert.pem</td>
<td>Client Certificate</td>
<td>certCli.pem</td>
</tr>
<tr>
<td>CliKey.pem</td>
<td>Client Key</td>
<td>privCli.pem</td>
</tr>
</tbody>
</table>

Once done that, the files can be transferred to the Client using for example an FTP client, connected with the TransPort router with usual username and password.

Please note that you may need to change your IP on the laptop accordingly with the new IP address configured on the ETH0 of the router.

In this example, in order to upload the files, the connection to the Transport is done on the local LAN (so using the ETH0 IP address of the router).
4.4 IKEv2/IPsec Tunnel Configuration

Phase 1 and 2 of the IKEv2/IPsec tunnel is configured as follows.

4.4.1 Phase 1 Settings

CONFIGURATION – NETWORK > VIRTUAL PRIVATE NETWORKING (VPN) > IPSEC > IKEv2 > IKEv2 0 and > ADVANCED

Where:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption</td>
<td>AES (256 bit)</td>
<td>The encryption algorithm used</td>
</tr>
<tr>
<td>Authentication</td>
<td>SHA1</td>
<td>The authentication algorithm used</td>
</tr>
<tr>
<td>PRF Algorithm</td>
<td>SHA1</td>
<td>The PRF (Pseudo Random Function) algorithm used</td>
</tr>
<tr>
<td>MODP Group for Phase 1</td>
<td>2 (1024)</td>
<td>Sets the key length used in the IKE Diffie-Hellman exchange to 768 bits (group 1) or 1024 bits (group 2). In this example group 2 is chosen to enable a 1024 bit key length</td>
</tr>
</tbody>
</table>
4.4.2 Phase 2 Settings

Then, the IPsec tunnel must be configured with the following settings:

CONFIGURATION – NETWORK > VIRTUAL PRIVATE NETWORKING (VPN) > IPSEC > IPSEC TUNNELS > IPSEC 0-9 > IPSEC 0
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>ToStrongSwan Server</td>
<td>Description of the IPsec tunnel</td>
</tr>
<tr>
<td>The IP address or hostname of the remote unit</td>
<td>10.104.1.125</td>
<td>The IP address or hostname of the remote IPsec peer that a VPN will be initiated to.</td>
</tr>
<tr>
<td>Local LAN IP Address</td>
<td>10.0.0.0</td>
<td>Use this IP address for the local LAN subnet. This is usually the IP address of the router’s Ethernet interface or that of a specific device on the local subnet</td>
</tr>
<tr>
<td>Local LAN Mask</td>
<td>255.255.255.0</td>
<td>Use this IP mask for the local LAN subnet. The mask sets the range of IP addresses that will be allowed to use the IPsec tunnel.</td>
</tr>
<tr>
<td>Remote LAN IP Address</td>
<td>10.0.1.0</td>
<td>Use this IP address for the remote LAN subnet. This is usually the IP address of the peer’s Ethernet interface or that of a specific device on the local subnet</td>
</tr>
<tr>
<td>Remote LAN Mask</td>
<td>255.255.255.0</td>
<td>Use this IP mask for the remote LAN subnet. The mask sets the range of IP addresses that will be allowed to use the IPsec tunnel.</td>
</tr>
<tr>
<td>Use the following security on this tunnel</td>
<td>RSA Signatures</td>
<td>Select RSA signature security for this tunnel to use the uploaded certificates</td>
</tr>
<tr>
<td>RSA Key File</td>
<td>privCli.pem</td>
<td>Private key file used for the responder</td>
</tr>
<tr>
<td>Our ID</td>
<td>client</td>
<td>ID that is matching the CN of the certificate in the first router (client)</td>
</tr>
<tr>
<td>Our ID type</td>
<td>IKE ID</td>
<td>Defines how the remote peer is to process the Our ID configuration. Set to IKE ID to match the information used in the certificate</td>
</tr>
<tr>
<td>Remote ID</td>
<td>server</td>
<td>Remote ID that is matching the CN in the second router certificate (server)</td>
</tr>
<tr>
<td>Use ( ) encryption on this tunnel</td>
<td>AES (256 bit keys)</td>
<td>The ESP encryption protocol to use with this IPsec tunnel</td>
</tr>
<tr>
<td>Use ( ) Authentication on this tunnel</td>
<td>SHA1</td>
<td>The ESP authentication algorithm to use with this IPsec tunnel</td>
</tr>
<tr>
<td>Use Diffie Hellman group ( )</td>
<td>No PFS</td>
<td>The Diffie Hellman (DH) group to use when negotiating new IPsec SAs.</td>
</tr>
<tr>
<td>Use IKE n to negotiate this tunnel</td>
<td>v2</td>
<td>The IKE version to use to negotiate this IPsec tunnel.</td>
</tr>
<tr>
<td>Use IKE configuration</td>
<td>0</td>
<td>The IKE configuration instance to use with this Eroute when the router is configured as an Initiator</td>
</tr>
<tr>
<td>Bring this tunnel up</td>
<td>All the time</td>
<td>This controls how the IPsec tunnel is brought up, for the initiator “All the time” option is chosen</td>
</tr>
<tr>
<td>If this tunnel is down and a packet is ready to be sent</td>
<td>Bring the tunnel up</td>
<td>Defines the action that is performed when the IPsec tunnel is down and a packet needs to be sent. For the initiator in this AN the “bring the tunnel up” option is chosen</td>
</tr>
</tbody>
</table>
5 CHECK IKEV2/IPSEC TUNNEL STATUS AND TEST

5.1 Check the IPsec tunnel is UP on TransPort WR

On TransPort WR the Event log will show the IKEv2 negotiation start and ends successfully:

MANAGEMENT – EVENT LOG

![Management - Event Log]

After that, in the connections status section IPsec and IKE v2 SAs will be displayed:

MANAGEMENT - CONNECTIONS > VIRTUAL PRIVATE NETWORKING (VPN) > IPSEC > IPSEC TUNNELS

![Management - Connections]

This can be checked also via CLI:

```
sastat
IPsec SAs (total:1). Eroute 0 -> 4
Outbound V1 SAs
  List Empty
Inbound V1 SAs
  List Empty
Outbound V2 SAs
  SPI Eroute  Peer IP  First Loc. IP  Last Loc. IP  TTL  KBytes
  Left        VIP
    c8552616  0    10.0.1.125               10.0.0.0        10.0.0.255  28713
    5d0df7eb  0    10.0.1.125               10.0.0.0        10.0.0.255  28713

Inbound V2 SAs
  SPI Eroute  Peer IP  First Loc. IP  Last Loc. IP  TTL  KBytes
  Left        VIP
    5d0df7eb  0    10.0.1.125               10.0.0.0        10.0.0.255  28713
    0         N/A                        N/A            N/A         N/A
```

OK
5.2 Check the IPsec tunnel is UP on StrongSwan

On The StrongSwan server, the IPsec tunnel status can be checked with the command “`sudo ipsec statusall`”:

digi@digi:~$ sudo ipsec statusall
[sudo] password for digi:
Status of IKE charon daemon (strongSwan 5.5.1, Linux 4.10.0-35-generic, x86_64):
  uptime: 2 hours, since Sep 25 12:35:49 2017
  malloc: sbrk 1486848, mmap 0, used 403040, free 1083808
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 5
  loaded plugins: charon test-vectors aes rc2 sha2 sha1 md4 md5 random nonce x509
  revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 ppp dnskey sshkey pem openssl
  fips-prf gmp agent xcbc hmac ccm gcm attr kernel-netlink resolve socket-default
  connmark stroke updown eap-mschapv2 xauth-generic
Listening IP addresses:
  10.104.1.125
  10.0.1.100
  10.8.0.1
Connections:
  DigiWR:  %any...%any IKEv2, dpddelay=300s
  DigiWR:  local:  [C=NL, O=Digi, CN=server] uses public key authentication
  DigiWR:  cert:  "C=NL, O=Digi, CN=server"
  DigiWR:  remote: uses public key authentication
  DigiWR:  child:  10.0.1.0/24 === 10.0.0.0/24 TUNNEL, dpdaction=clear
Security Associations (1 up, 0 connecting):
  DigiWR[8]:  ESTABLISHED 4 minutes ago, 10.104.1.125[C=NL, 0=Digi, CN=server]...10.104.1.115[C=NL, O=Digi, CN=client]
  DigiWR[8]:  IKEv2 SPIs: 00000107fffffe8_i 743edad58684761_r*, public key reauthentication in 2 hours
  DigiWR[5]:  IKE proposal: AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024
  DigiWR[5]:  INSTALLED, TUNNEL, reqid 2, ESP SPIs: c8552c16_i 5d0df7eb_o
  DigiWR[5]:  AES_CBC_256/HMAC_SHA1_96, 0 bytes_i, 0 bytes_o, rekeying in 39 minutes
  DigiWR[5]:  10.8.1.0/24 === 10.0.0.0/24
5.3 Testing

An easy way to test if the LAN to LAN traffic pass through the tunnel, is to generate a ping from each side of the tunnel to reach the remote end's local interface.

From TransPort WR:

![Ping results from TransPort WR]

From StrongSwan side:

digi@digi:~$ ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=250 time=1.47 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=250 time=1.48 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=250 time=1.23 ms
6 CONFIGURATION FILES

6.1 TransPort WR configuration file

The TransPort WR configuration file used for the purpose of this Application Note is shown below:

```plaintext
config c show
eth 0 IPaddr "10.0.0.1"
lapb 0 ans OFF
lapb 1 tinact 120
lapb 3 dtemode 0
lapb 4 dtemode 0
lapb 5 dtemode 0
lapb 6 dtemode 0
ip 0 cidr ON
def_route 0 ll_ent "PPP"
def_route 0 ll_add 1
eroute 0 descr "ToStrongSwan Server"
eroute 0 peerip "10.104.1.125"
eroute 0 peerid "server"
eroute 0 ourid "client"
eroute 0 locip "10.0.0.0"
eroute 0 locmsk "255.255.255.0"
eroute 0 remip "10.0.1.0"
eroute 0 remmsk "255.255.255.0"
eroute 0 ESPauth "SHA1"
eroute 0 ESPenc "AES"
eroute 0 authmeth "RSA"
eroute 0 nosa "TRY"
eroute 0 autosa 2
route 0 ikever 2
route 0 enckeybits 256
route 0 privkey "privCli.pem"
dhcp 0 respdelms 500
dhcp 0 mask "255.255.255.0"
dhcp 0 gateway "192.168.1.1"
dhcp 0 DNS "192.168.1.1"
sntp 0 server "time.devicecloud.com"
dyndns 0 ifent "default"
ppp 0 timeout 300
ppp 1 name "W-WAN (LTE)"
ppp 1 phonenum "*98*1#"
ppp 1 IPaddr "0.0.0.0"
ppp 1 timeout 0
ppp 1 ipsec 1
ppp 1 use_modem 1
ppp 1 aodion 1
ppp 1 autoassert 1
ppp 1 r_chap OFF
ppp 3 defpak 16
ppp 4 defpak 16
web 0 prelogin_info ON
web 0 showgswiz ON
ike 0 deblevel 4
ike2 0 iencalg "AES"
ike2 0 ienckeybits 256
ike2 0 idhgroup 2
modemcc 0 info_asy_add 4
```
modemcc 0 init_str "+CGQREQ=1"
modemcc 0 init_str1 "+CGQMIN=1"
modemcc 0 apn "Your.APN.goes.here"
modemcc 0 link_retries 10
modemcc 0 stat_retries 30
modemcc 0 sms_interval 1
modemcc 0 sms_access 1
modemcc 0 sms_concat 0
modemcc 0 init_str_2 "+CGQREQ=1"
modemcc 0 init_str1_2 "+CGQMIN=1"
modemcc 0 apn_2 "Your.APN.goes.here"
modemcc 0 link_retries_2 10
modemcc 0 stat_retries_2 30
modemcc 0 sms_access_2 1
modemcc 0 sms_concat_2 0
ana 0 anon ON
ana 0 l2on OFF
ana 0 l3on OFF
ana 0 xoton OFF
ana 0 lapdon 0
ana 0 lapbon 0
ana 0 maxdata 1500
ana 0 logsize 180
cmd 0 unitid "ss%s>
cmd 0 cmdnua "99"
cmd 0 hostname "digi.router"
cmd 0 asyled_mode 2
cmd 0 tremto 1200
cmd 0 rcihttp ON
user 0 access 0
user 1 name "username"
user 1 epassword "KD51SVJDVVg="
user 1 access 0
user 2 access 0
user 3 access 0
user 4 access 0
user 5 access 0
user 6 access 0
user 7 access 0
user 8 access 0
user 9 access 0
local 0 transaccess 2
sslcli 0 verify 10
sslsrvr 0 certfile "cert01.pem"
sslsrvr 0 keyfile "privrsa.pem"
ssh 0 hostkey1 "privSSH.pem"
ssh 0 nb_listen 5
ssh 0 v1 OFF
templog 0 mo_autooff ON
cloud 0 ssl ON

Power Up Profile: 0
OK
6.2 StrongSwan Server configuration files

The StrongSwan configuration files used for the purpose of this Application Note are shown below:

ipsec.conf

# ipsec.conf - strongSwan IPsec configuration file
# basic configuration

config setup

# Add connections here.

conn %default
  keyexchange=ikev2
  ike=aes256-sha1-modp1024
  esp=aes256-sha1
  dpdaction=clear
  dpddelay=300s

conn DigiWR
  left=%any
  leftcert=SrvCert.pem
  leftsubnet=10.0.1.0/24
  right=%any
  rightsubnet=10.0.0.0/24
  auto=add

ipsec.secrets

# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.

: RSA SrvKey.pem