EC2x&EG9x&EG25-G Series
QuecOpen Linux VPN Porting Guide

LTE Standard Module Series

Version: 1.0

Date: 2020-08-13

Status: Released

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236 Email: info@quectel.com

Or our local office. For more information, please visit: http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm or email to support@quectel.com.

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

DISCLAIMER

WHILE QUECTEL HAS MADE EFFORTS TO ENSURE THAT THE FUNCTIONS AND FEATURES UNDER DEVELOPMENT ARE FREE FROM ERRORS, IT IS POSSIBLE THAT THESE FUNCTIONS AND FEATURES COULD CONTAIN ERRORS, INACCURACIES AND OMISSIONS. UNLESS OTHERWISE PROVIDED BY VALID AGREEMENT, QUECTEL MAKES NO WARRANTIES OF ANY KIND, IMPLIED OR EXPRESS, WITH RESPECT TO THE USE OF FEATURES AND FUNCTIONS UNDER DEVELOPMENT. TO THE MAXIMUM EXTENT PERMITTED BY LAW, QUECTEL EXCLUDES ALL LIABILITY FOR ANY LOSS OR DAMAGE SUFFERED IN CONNECTION WITH THE USE OF THE FUNCTIONS AND FEATURES UNDER DEVELOPMENT, REGARDLESS OF WHETHER SUCH LOSS OR DAMAGE MAY HAVE BEEN FORESEEABLE.

COPYRIGHT

THE INFORMATION CONTAINED HERE IS PROPRIETARY TECHNICAL INFORMATION OF QUECTEL WIRELESS SOLUTIONS CO., LTD. TRANSMITTING, REPRODUCING, DISSEMINATING AND EDITING THIS DOCUMENT AS WELL AS USING THE CONTENT WITHOUT PERMISSION ARE FORBIDDEN. OFFENDERS WILL BE HELD LIABLE FOR PAYMENT OF DAMAGES. ALL RIGHTS ARE RESERVED IN THE EVENT OF A PATENT GRANT OR REGISTRATION OF A UTILITY MODEL OR DESIGN.

About the Document

Revision History

Version	Date	Author	Description
1.0	2020-08-13	Tinker SUN	Initial

Contents

Table Index

1 Introduction

Quectel LTE Standard EC2x&EG9x&EG25-G Series modules support QuecOpen®. This document mainly describes how to port VPN software which is applied to Linux system to the module. There is no difference between porting VPN software in QuecOpen® and porting the third-party open source software in other cross-compilation environments. Hence you can refer to this document to port other open source software in QuecOpen®.

Before porting VPN software, confirm that QuecOpen® compilation environment has been built.

1.1. Applicable Modules

Table 1: Applicable Modules

Module Series	Module
EC2x series	EC25 series
	EC21 series
	EC20 R2.1
EG9x series	EG95 series
	EG91 series
EG25-G	EG25-G

2 Porting Process

2.1. Download Source Code

According to your needs, download related VPN software source code: pptp-linux, xl2tpd, pppd, gmp and strongswan. The homepage, version and download address are as follows:

1) PPTP
- Software: pptp-linux
- Homepage: http://pptpclient.sourceforge.net/
- Version: 1.10.0
- Download address: *https://nchc.dl.sourceforge.net/project/pptpclient/pptp/pptp-1.10.0/pptp-1.10.0.tar.gz*

2) L2TP
- Software: xl2tpd
- Homepage: https://www.xelerance.com/archives/155
- Version: 1.3.11
- Download address: *https://codeload.github.com/xelerance/xl2tpd/zip/1.3.11*

3) PPP
- Software: pppd
- Homepage: https://download.samba.org/
- Version: 2.4.7
- Download address: *https://download.samba.org/pub/ppp/ppp-2.4.7.tar.gz*

4) GMP
- Software: gmp
- Homepage: https://gmplib.org/
- Version: 6.1.2
- Download address: *https://gmplib.org/download/gmp/gmp-6.1.2.tar.bz2*

5) IPSEC
- Software: strongswan
- Homepage: https://www.strongswan.org/
- Version: 5.6.2
- Download address: *https://download.strongswan.org/strongswan-5.6.2.tar.bz2*

2.2. Preparation for Compilation

Step 1: Create new *opensrc* directory in the *ql-ol-sdk* directory, and place the downloaded source files in the *opensrc* directory.

```
mkdir opensrc
```

Step 2: Create *Makefile* file in the *opensrc* directory and input the contents as follows:

```
CURR_DIR := $(shell pwd)
OPENSRC_DIR := $(subst /opensrc, /opensrc, $(CURR_DIR))
OPENSRC_DIR := $(word 1, $(WORKSPACE_DIR))

export PKG_CONFIG_SYSROOT_DIR=$(SDKTARGETSYSROOT)
export PKG_CONFIG_PATH=$(SDKTARGETSYSROOT)/usr/lib/pkgconfig

BUILD_DESTDIR=$(CURR_DIR)/rootfs
BUILD_HOST=arm-oe-linux-gnueabi
BUILD_TARGET=arm-oe-linux-gnueabi

targets_build=pppd_build pptp_build xl2tpd_build libgmp_build strongswan_build
targets_clean=pppd_clean pptp_clean xl2tpd_clean libgmp_clean strongswan_clean

CFLAGS+=-I$(BUILD_DESTDIR)/include -I$(BUILD_DESTDIR)/usr/include
LDFLAGS+=-L$(BUILD_DESTDIR)/lib -L$(BUILD_DESTDIR)/usr/lib

SRC_PPPD:=ppp-2.4.7
SRC_GMP:=gmp-6.1.2
SRC_STRONGWAN=strongswan-5.6.2
SRC_XL2TPD=xl2tpd-1.3.11
SRC_PPTP=pptp-1.10.0

.PHONY: all
all: $(targets_build)
	rm -rf rootfs_build;
	cp -arf rootfs rootfs_build;
	rm -rf rootfs_build/include rootfs_build/usr/include rootfs_build/share rootfs_build/usr/share;
	find rootfs_build -name "*.a" | xargs rm -f
	@for ff in $(shell find rootfs_build -type f); do \
		$(STRIP) $$ff 2>/dev/null && echo "STRIP FILE :" $$ff;\
	done
	@echo "======================compile $(targets) complete======================"
```

```
clean: $(targets_clean)
	rm -rf $(BUILD_DESTDIR)
	@echo "======================clean $(targets) complete======================"

pptp_build:
	if [ ! -d $(SRC_PPTP) ]; then \
		tar xkf $(SRC_PPTP).tar.gz 2>/dev/null; \
	fi
	cd $(SRC_PPTP) && make DESTDIR=$(BUILD_DESTDIR) CC="$(CC)" IP="/sbin/ip"
PPPD="/usr/sbin/pppd" && \
	fakeroot make install DESTDIR=$(BUILD_DESTDIR)
	@echo "compile $(SRC_PPTP) completed"

pptp_clean:
	if [ -d $(SRC_PPTP) ]; then \
		cd $(SRC_PPTP) && make clean; \
	fi

xl2tpd_build:
	if [ ! -d $(SRC_XL2TPD) ]; then \
		unzip -n $(SRC_XL2TPD).zip 2>/dev/null; \
	fi
	cd $(SRC_XL2TPD) && make PREFIX=$(BUILD_DESTDIR) && \
	make install PREFIX=$(BUILD_DESTDIR)
	@echo "compile $(SRC_XL2TPD) completed"

xl2tpd_clean:
	if [ -d $(SRC_XL2TPD) ]; then \
		cd $(SRC_XL2TPD) && make clean; \
	fi

pppd_build:
	if [ ! -e $(SRC_PPPD)/Makefile ]; then \
		tar xkf $(SRC_PPPD).tar.gz 2>/dev/null; \
		cd $(SRC_PPPD); \
		./configure \
		INSTROOT="$(BUILD_DESTDIR)" \
		DESTDIR="$(BUILD_DESTDIR)" \
		BINDIR=$(BUILD_DESTDIR)/usr/sbin; \
	fi
	cd $(SRC_PPPD) && make && make INSTROOT="$(BUILD_DESTDIR)" \
		INSTALL="install --strip-program=$(STRIP)" \
		DESTDIR="$(BUILD_DESTDIR)" \
		BINDIR=$(BUILD_DESTDIR)/usr/sbin install
```

```
    @echo "compile $(SRC_PPPD) completed"

pppd_clean:
    if [ -e $(SRC_PPPD)/Makefile ]; then \
        cd $(SRC_PPPD) && make clean;\
    fi

libgmp_build:
    if [ ! -e $(SRC_GMP)/Makefile ]; then \
        tar xkf $(SRC_GMP).tar.bz2 2>/dev/null; \
        cd $(SRC_GMP); \
        ./configure \
        --host=$(BUILD_HOST) \
        --target=$(BUILD_TARGET) \
        --prefix=$(BUILD_DESTDIR) \
        --disable-silent-rules \
        --disable-dependency-tracking \
        --enable-cxx=detect \
        --with-readline=no; \
    fi
    cd $(SRC_GMP) && make && make install
    @echo "compile $(SRC_GMP) completed"

libgmp_clean:
    if [ -e $(SRC_GMP)/Makefile ]; then \
        cd $(SRC_GMP) && make clean; \
    fi


strongswan_build:
    if [ ! -e $(SRC_STRONGWAN)/Makefile ]; then \
        tar xkf $(SRC_STRONGWAN).tar.bz2 2>/dev/null; \
        cd $(SRC_STRONGWAN); \
        ./configure \
        --host=$(BUILD_HOST) \
        --target=$(BUILD_TARGET) \
        --prefix=/ \
        --disable-silent-rules \
        --disable-dependency-tracking \
        --without-lib-prefix    \
        --without-systemdsystemunitdir \
        --disable-aesni \
        --enable-charon \
        --enable-curl \
```

```
            --enable-gmp \
            --enable-eap-md5 \
            --disable-ldap \
            --disable-mysql \
            --enable-openssl \
            --disable-scepclient \
            --disable-soup \
            --enable-sqlite \
            --enable-stroke \
            --disable-swanctl \
            --disable-systemd \
            CFLAGS="$(CFLAGS)" \
            LDFLAGS="$(LDFLAGS)"; \
    fi
    cd $(SRC_STRONGWAN) && make && make install DESTDIR=$(BUILD_DESTDIR)
    @echo "compile $(SRC_STRONGWAN) completed"

strongswan_clean:
    if [ -e $(SRC_STRONGWAN)/Makefile ]; then \
        cd $(SRC_STRONGWAN) && make clean; \
    fi
```

The final directory structure of *ql-ol-sdk* directory is as follows:

```
ql-ol-sdk
├─── Makefile
├─── opensrc
│    ├─── gmp-6.1.2.tar.bz2
│    ├─── Makefile
│    ├─── ppp-2.4.7.tar.gz
│    ├─── pptp-1.10.0.tar.gz
│    ├─── strongswan-5.6.2.tar.bz2
│    └─── xl2tpd-1.3.11.zip
├─── ql-ol-bootloader
├─── ql-ol-crosstool
├─── ql-ol-extsdk
├─── ql-ol-kernel
├─── ql-ol-rootfs
├─── ql-ol-usrdata
├─── ql-ol-usrfs
└─── target
```

2.3. Modify Kernel Compilation Option

Modify the kernel compilation option to port the VPN software, taking modifying the kernel compilation option files of PPTP, L2TP and IPSEC as an example. The path of the files is *ql-ol-kernel/msm-3.18/arch/arm/configs/mdm9607-perf_defconfig.* The modified kernel compilation options are as follows:

```
PPTP:
CONFIG_PPP_MPPE=y

L2TP:
CONFIG_NET_UDP_TUNNEL=y
CONFIG_L2TP=y

IPSEC:
CONFIG_INET_AH=m
CONFIG_INET_ESP=m
CONFIG_INET_IPCOMP=m
CONFIG_INET_XFRM_TUNNEL=m
CONFIG_INET_TUNNEL=m
CONFIG_XFRM_USER=m
```

2.4. Compile Files

PPTP and L2TP depend on PPP-related plugins which should be recompiled before using PPTP and L2TP. IPSEC uses strongswan that depends on library libgmp which should be compiled before strongswan compilation. Take compiling strongwan as an example:

1. Strongswan needs to use m4 tool in the process of compilation. Taking the Ubuntu environment as an example, Ubuntu runs the following commands to install m4 tool:

```
sudo apt install m4
```

2. Enter the *ql-ol-sdk* directory and run the following commands to compile

```
source ql-ol-crosstool/ql-ol-crosstool-env-init
make kernel
make kernel_module
cd opensrc
make
```

3. After compilation, the target file is in the path of *opensrc/rootfs_build.*

2.5. Pack the Generated Files to Rootfs

After editing the *makefile* in the *ql-ol-sdk* directory, execute the **make rootfs** command to pack the compiled files in the *opensrc* directory to rootfs in the path of *ql-ol-sdk/ql-ol-rootfs*. The following is taking the SDK used in the module as an example to explain how to edit the *makefile*.

Before modification:

```
$(rootfs):
ifneq ($(filter $(QUECTEL_PROJECT_NAME), AG35C AG35CE), )
    cd $(TOPDIR) ; chmod +x ./ql-ol-extsdk/tools/quectel_ubi/* ; ./ql-ol-extsdk/tools/quectel_ubi/mkfs.ubi
fs -r ql-ol-rootfs -o machine-image-mdm9610.ubifs -m 2048 -e 126976 -c 4292 -F    ; \
    ./ql-ol-extsdk/tools/quectel_ubi/mkfs.ubifs -r ql-ol-usrfs -o mdm9607-usrfs.ubifs -m 2048 -e 126976 -c
 4292 -F ;    \
```

After modification:

```
    cd $(TOPDIR) ; chmod +x ./ql-ol-extsdk/tools/quectel_ubi/* ; \
    cp -arf ql-ol-rootfs temprootfs; \
    cp -arf opensrc/rootfs_build/* temprootfs/ ; \
    ./ql-ol-extsdk/tools/quectel_ubi/mkfs.ubifs -r temprootfs -o machine-image-mdm9610.ubifs -m 2048 -e 12
6976 -c 4292 -F   ; \
    rm -rf temprootfs; \
    ./ql-ol-extsdk/tools/quectel_ubi/mkfs.ubifs -r ql-ol-usrfs -o mdm9607-usrfs.ubifs -m 2048 -e 126976 -c
 4292 -F ;    \
```

3 Software Test

This chapter describes how to test VPN software in QuecOpen to confirm whether the software is successfully ported, taking pptp-linux, xl2tpd and strongswan as an example. The protocols used by the above software are PPTP, L2TP, and IPSEC.

3.1. PPTP

The following example illustrates how to use the authentication user name (test) and password (11111111) to connect to the PPTP server configuration (192.168.20.49).

Step 1: Edit the file in the path of */etc/ppp/chap-secrets*, and add authentication user name and password.

```
test * 11111111 *
```

Step 2: Edit the file in the path of */etc/ppp/peers/pptpvpn.*

```
pty "pptp 192.168.20.49 --nolaunchpppd"
lock
noauth
nobsdcomp
nodeflate
name test
remotename pptpvpn
ipparam pptpvpn
require-mppe-128
```

Step 3: start dialing:

```
pppd call pptpvpn updetach
```

Step 4: If the dial status is as follows, it indicates that the porting is successful.

```
/etc/ppp/peers # ifconfig ppp0
ppp0      Link encap:Point-to-Point Protocol
          inet addr:192.168.20.26  P-t-P:192.168.20.230  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1396  Metric:1
          RX packets:30 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:2509 (2.4 KiB)  TX bytes:90 (90.0 B)
```

3.2. L2TP

The following example illustrates how to use authentication username (test) password (11111111) to connect to L2TP server configuration (192.168.20.49).

Step 1: Edit xl2tpd configuration file in the path of */etc/xl2tpd/xl2tpd.conf*:

```
[global]
port = 1701
debug state = yes
debug tunnel = yes

[lac testvpn]
lns = 192.168.20.49
require chap = yes
refuse pap = yes
require authentication = yes
name = test
ppp debug = yes
pppoptfile = /etc/ppp/peers/testvpn.l2tpd
length bit = yes
```

Step 2: Create the L2TP connection configuration file specified in the file that is in the path of */etc/xl2tpd/xl2tpd.conf,* and the created file is in path of */etc/PPP/peers/testvpn.l2tpd.*

```
user test
password 11111111
noauth
lock
lcp-echo-interval 3
lcp-echo-failure 30
asyncmap 0
```

Step 3: Start dialing:

```
mkdir /var/run/xl2tpd
xl2tpd –D &
echo "c testvpn" > /var/run/xl2tpd/l2tp-control
```

Step 4: If the dial status is as follows, it indicates that the porting is successful.

```
ppp0      Link encap:Point-to-Point Protocol
          inet addr:192.168.20.124  P-t-P:192.168.20.230  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1400  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:226 (226.0 B)  TX bytes:64 (64.0 B)
```

3.3. IPSEC

The test configuration about strongswan can refer to https://www.strongswan.org/testresults.html. Use ikev1/net2net-psk scenario to test. The server (192.168.10.154) is a host running strongswan on the WAN side, and the client (192.168.22.17) is our module.

3.3.1. Server configuration (192.168.10.154)

Step 1: Edit the file in the path of */etc/ipsec.conf.*

```
config setup

conn %default
    ikelifetime=60m
    keylife=20m
    rekeymargin=3m
    keyingtries=1
    keyexchange=ikev2
    authby=secret

conn net-net
    left=192.168.10.154
    leftid=@moon.strongswan.org
    right=192.168.22.17
    rightid=@sun.strongswan.org
    auto=add
```

Step 2: Edit the file in the path of */etc/ipsec.secrets*.

```
@moon.strongswan.org @sun.strongswan.org : PSK 0sv+NkxY9LLZvwj4qCC2o/gGrWDF2d21jL
```

Step 3: Edit the file in the path of */etc/strongswan.conf*:

```
charon {
    load = random nonce aes sha1 sha2 curve25519 hmac stroke kernel-netlink socket-default updown
}
```

Step 4: Start the strongswan service:

```
sudo ipsec start --nofork --debug-all
```

3.3.2. Module configuration (192.168.22.17)

Step 1: Edit the file in the path of */etc/ipsec.conf.*

```
config setup

conn %default
        ikelifetime=60m
        keylife=20m
        rekeymargin=3m
        keyingtries=1
        keyexchange=ikev2
        authby=secret

conn net-net
        left=192.168.22.17
        leftid=@sun.strongswan.org
        leftfirewall=yes
        right=192.168.10.154
        rightid=@moon.strongswan.org
        auto=add
```

Step 2: Edit the file in the path of */etc/ipsec.secrets.*

```
@moon.strongswan.org                    @sun.strongswan.org                    PSK
0sv+NkxY9LLZvwj4qCC2o/gGrWDF2d21jL
```

Step 3: Edit the file in the path of */etc/strongswan.conf.*

```
charon {
   load = random nonce aes sha1 sha2 curve25519 hmac stroke kernel-netlink socket-default updown
}
```

Step 4: Start dialing:

```
/etc # ipsec start
Starting strongSwan 5.6.2 IPsec [starter]...
!! Your strongswan.conf contains manual plugin load options for charon.
!! This is recommended for experts only, see
!! http://wiki.strongswan.org/projects/strongswan/wiki/PluginLoad
/etc # ipsec up net-net
initiating IKE_SA net-net[1] to 192.168.10.154
generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(FRAG_SUP) N(HASH_ALG) N(REDIR_SUP) ]
sending packet: from 192.168.22.17[500] to 192.168.10.154[500] (312 bytes)
received packet: from 192.168.10.154[500] to 192.168.22.17[500] (240 bytes)
parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(FRAG_SUP) N(HASH_ALG) N(MULT_AUTH) ]
local host is behind NAT, sending keep alives
authentication of 'sun.strongswan.org' (myself) with pre-shared key
establishing CHILD_SA net-net{1}
generating IKE_AUTH request 1 [ IDi N(INIT_CONTACT) IDr AUTH SA TSi TSr N(MOBIKE_SUP) N(ADD_4_ADDR) N(ADD_4_ADDR
sending packet: from 192.168.22.17[4500] to 192.168.10.154[4500] (384 bytes)
received packet: from 192.168.10.154[4500] to 192.168.22.17[4500] (256 bytes)
parsed IKE_AUTH response 1 [ IDr AUTH SA TSi TSr N(AUTH_LFT) N(MOBIKE_SUP) N(NO_ADD_ADDR) ]
authentication of 'moon.strongswan.org' with pre-shared key successful
IKE_SA net-net[1] established between 192.168.22.17[sun.strongswan.org]...192.168.10.154[moon.strongswan.org]
scheduling reauthentication in 3353s
maximum IKE_SA lifetime 3533s
CHILD_SA net-net{1} established with SPIs c0e244a8_i c6b9e7cf_o and TS 192.168.22.17/32 === 192.168.10.154/32
connection 'net-net' established successfully
```

Step 5: If the dial status is as follows, it indicates that the migration is successful.

```
/ # ipsec status
Security Associations (1 up, 0 connecting):
     net-net[1]: ESTABLISHED 15 seconds ago, 192.168.22.17[sun.strongswan.org]...192.168.10.154[moon.strongswan.org]
     net-net{1}:  INSTALLED, TUNNEL, reqid 1, ESP in UDP SPIs: c3e576f1_i cd50a876_o
     net-net{1}:   192.168.22.17/32 === 192.168.10.154/32
/ #
```

```
/etc # ip -s xfrm state
src 192.168.22.17 dst 192.168.10.154
        proto esp spi 0xc6b9e7cf(3334072271) reqid 1(0x00000001) mode tunnel
        replay-window 0 seq 0x00000000 flag af-unspec (0x00100000)
        auth-trunc hmac(sha256) 0xab1ca4f01aacb4d4b9674950bea3397c82748659a6586591926b9427eea13a0c (256 bits) 12
        enc cbc(aes) 0x768ccdaf03ea112f160007c82985fe87 (128 bits)
        encap type espinudp sport 4500 dport 4500 addr 0.0.0.0
        lifetime config:
          limit: soft (INF)(bytes), hard (INF)(bytes)
          limit: soft (INF)(packets), hard (INF)(packets)
          expire add: soft 944(sec), hard 1200(sec)
          expire use: soft 0(sec), hard 0(sec)
        lifetime current:
          0(bytes), 0(packets)
          add 2018-03-04 08:54:15 use -
        stats:
          replay-window 0 replay 0 failed 0
src 192.168.10.154 dst 192.168.22.17
        proto esp spi 0xc0e244a8(3236054184) reqid 1(0x00000001) mode tunnel
        replay-window 32 seq 0x00000000 flag af-unspec (0x00100000)
        auth-trunc hmac(sha256) 0x5c7e11f1aac4427f472996b9f5cbac6ddde3b4319b7103d921835db11879952a (256 bits) 12
        enc cbc(aes) 0xea80a18be09f76c8cc097fbfc1647ca7 (128 bits)
        encap type espinudp sport 4500 dport 4500 addr 0.0.0.0
        lifetime config:
          limit: soft (INF)(bytes), hard (INF)(bytes)
          limit: soft (INF)(packets), hard (INF)(packets)
          expire add: soft 954(sec), hard 1200(sec)
          expire use: soft 0(sec), hard 0(sec)
        lifetime current:
          0(bytes), 0(packets)
          add 2018-03-04 08:54:15 use -
        stats:
          replay-window 0 replay 0 failed 0
```

4 Appendix A References

Table 2: Terms and Abbreviations

Abbreviation	Description
GMP	GNU Multiple Precision Arithmetic Library
IPSEC	Internet Protocol Security
L2TP	Layer-2 Tunneling Protocol
PPTP	Point-to-Point Tunneling Protocol
PPP	Point to Point Protocol
SDK	Software Development Kit
VPN	Virtual Private Network
WAN	Wide Area Network