PPP Protocol Modules for TTCN-3 Toolset with TITAN, User Guide

János Kövesdi

Version 198 17-CNL 113 599, Rev. B, 2013-09-02

Table of Contents

About This Document

How to Read This Document

This is the User Guide for the PPP protocol module. The PPP protocol module is developed for the TTCN-3 Toolset with TITAN. This document should be read together with Function Specification [4].

Presumed Knowledge

To use this protocol module the knowledge of the TTCN-3 language [1] is essential.

System Requirements

Protocol modules are a set of TTCN-3 source code files that can be used as part of TTCN-3 test suites only. Hence, protocol modules alone do not put specific requirements on the system used. However in order to compile and execute a TTCN-3 test suite using the set of protocol modules the following system requirements must be satisfied:

- TITAN TTCN-3 Test Executor installed. For installation guide see [3].

Protocol Modules

Overview

Protocol modules implement the message structures of the corresponding protocol in a formalized way, using the standard specification language TTCN-3. This allows defining of test data (templates) in the TTCN-3 language [1] and correctly encoding/decoding messages when executing test suites using the TITAN TTCN-3 test environment.

Protocol modules are using TITAN's RAW encoding attributes [2] and hence are usable with the TITAN test toolset only.

The PPP messages are represented by the TTCN-3 record type `PDU_PPP`.

The information field of PDU_PPP is a TTCN-3 union which contains the appropriate associated subtype IP, LCP, IPCP, CHAP, PAP or EAP.

The implemented IP messages in TTCN-3 are defined in [14].

The implemented LCP messages in TTCN-3 are:

`LCP_ConfigureRequest`
`LCP_ConfigureAck`
`LCP_ConfigureNak`
`LCP_ConfigureReject`

LCP_ConfigureRequest
LCP_TerminateRequest
LCP_TerminateAck
LCP_CodeReject
LCP_ProtocolReject
LCP_EchoRequest
LCP_EchoReply
LCP_DiscardRequest

The implemented IPCP messages in TTCN-3 are:

IPCP_ConfigureRequest
IPCP_ConfigureAck
IPCP_ConfigureNak
IPCP_ConfigureReject
IPCP_TerminateRequest
IPCP_TerminateAck
IPCP_CodeReject

The implemented CHAP messages in TTCN-3 are:

CHAP_Challenge
CHAP_Response
CHAP_Success
CHAP_Failure

The implemented PAP messages in TTCN-3 are:

PAP_AuthenticateRequest
PAP_AuthenticateAck
PAP_AuthenticateNak

Using these type records, templates can be defined to send and receive a given message.

The *EAP_Types.ttcn* module contains the implemented PPP EAP messages. These messages have the same structure as these have in the RADIUS Test Port:

eap_identity
eap_notification
eap_nak
eap_md5_challenge
eap_one_time_password
eap_generic_token_card
eap_sim
eap_aka

Installation

The set of protocol modules can be used in developing TTCN-3 test suites using any text editor. However to make the work more efficient a TTCN-3-enabled text editor is recommended (for example, `nedit`, `xemacs`). Since the PPP protocol module is used as a part of a TTCN-3 test suite, this requires TTCN-3 Test Executor and a C compiler be installed before the module can be compiled and executed together with other parts of the test suite. For more details on the installation of TTCN-3 Test Executor see the relevant parts of [2].

Configuration

Compile-Time configuration

The compile-time configuration of the RADIUS test port is performed by customizing the generated *Makefile*. The following steps must be made:

1. The `OPENSSL_DIR` variable must be set to the location of OpenSSL.
2. The `CPPFLAGS` must contain `-I$(OPENSSL_DIR)/include`
3. The operation specific libraries (for example, `SOLARIS_LIBS`) should contain `-lresolv`.

Runtime Configuration

The behavior of the executable test program is determined by the run-time configuration file. This is a simple text file, which contains various sections (e.g. `[TESTPORT_PARAMETERS]`) after each other. The usual suffix of configuration files is *.cfg*. Only the `[MODULE_PARAMETERS]` section is related to the PPP (EAP) protocol module. In this section you can specify parameters that are passed to the protocol module.

The following parameters are allowed:

- `tsp_skip_auth_encr` (OPTIONAL): If this parameter is set to `true`, the authentication and encryption functionality of EAP-SIM and EAP-AKA is turned off.
- `tsp_global_keying` (OPTIONAL): If it is set to `false`, the protocol module uses EAP-Identifier based keying material for EAP-SIM and EAP-AKA, ie., keying material is treated separately for each EAP-Identifier. If this parameter is set to `true`, then the test port uses global keying with EAP-SIM and EAP-AKA (a pseudo-value 256 is used as EAP-Identifier).

  The default value is `false`.

- `tsp_debugging` (OPTIONAL): This boolean parameter allows the output of textual debug information of TTCN-3 "log" statements on the console or in log file (depending on the setting of `consoleMask` and `fileMask` parameters).

NOTE	Error messages for serious errors are not affected by the `tsp_debugging` parameter. The default value is `false`.

- `tsp_SIM_Ki` (OPTIONAL): The length of this octetstring parameter is `16` octet. `Ki` key has to be set prior to sending or receiving EAP-SIM messages containing `AT_ENCR_DATA`. The `Ki` key will be set automatically with `tsp_SIM_Ki` if it is not set with `f_set_Ki` function.

  The default value is `'00112233445566778899AABBCCDDEEFF'O`

- `tsp_AKA_K` (OPTIONAL): The length of this octetstring parameter is `16` octet. `K` key has to be set prior to sending or receiving EAP-AKA messages containing `AT_ENCR_DATA`. The `AKA K` key will be set automatically with `tsp_AKA_K` if it is not set with `f_set_K` function.

  The default value is `'0123456789ABCDEF0123456789ABCDEF'O`

- `tsp_AKA_SQN` (OPTIONAL): The length of this octetstring parameter is `6` octet. `SQN` key has to be set prior to sending EAP-AKA messages containing `AT_AUTN`. The `AKA SQN` key will be set automatically with `tsp_AKA_SQN` if it is not set with `f_set_SQN` function.

  The default value is `'000000000000'O`

- `tsp_AKA_SQN_MS` (OPTIONAL): The length of this octetstring parameter is `6` octet. `SQN_MS` key has to be set prior to sending EAP-AKA messages containing `AT_AUTS`. The `AKA SQN_MS` key will be set automatically with `tsp_AKA_SQN_MS` if it is not set with `f_set_SQN_MS` function.

  The default value is `'000000000000'O`

- `tsp_AKA_AMF` (OPTIONAL): The length of this octetstring parameter is `2` octet. `AMF` key has to be set prior to sending EAP-AKA messages containing `AT_AUNT`. The `AKA AMF` key will be set automatically with `tsp_AKA_AMF` if it is not set with `f_set_AMF` function.

  The default value is `'0000'O`

Encoder and Decoder Functions

The following encoder/decoder functions are available which provide for the correct encoding of messages when sent from TITAN and correct decoding of messages when received by TITAN.:

Name	Type of formal parameters	Type of return value
`enc_PDU_PPP`	PDU_PPP	octetstring
`dec_PDU_PPP`	octetstring	PDU_PPP

NOTE	The Address and Control fields defined in [10] are treated as a single optional field at the beginning of `PDU_PPP`.

PPP EAP Functions

Encoder and Decoder Functions

Name	Type of parameters	Type of return value
`f_enc_PDU_EAP`	PDU_EAP	octetstring
`f_dec_PDU_EAP`	octetstring	PDU_EAP
`f_enc_PDU_EAP_list`	PDU_EAP_list	octetstring
`f_dec_PDU_EAP_list`	octetstring	PDU_EAP_list
`f_enc_eap_sim_attrib_list`	eap_sim_attrib_list	octetstring
`f_dec_eap_sim_attrib_list`	octetstring	eap_sim_attrib_list
`f_enc_eap_aka_attrib_list`	eap_aka_attrib_list	octetstring
`f_dec_eap_aka_attrib_list`	octetstring	eap_aka_attrib_list

Authentication and Encryption Key Generation

`EAP_port_descriptor` stores the authentication and encryption keys. It is needed to be initialized; during the use of a descriptor variable without initialization can occur errors!

```
function f_initEAPPortDescriptor(inout EAP_port_descriptor descriptor);
```

Function for automatic generation and storage of authentication and encryption keys:

```
function f_get_EAP_parameters(inout octetstring pl_ext_eap_message,inout
EAP_port_descriptor pl_descriptor,in boolean incoming_message)
```

Function for generating AT_MAC, Kaut key is needed:

```
function f_calc_HMAC(in octetstring key, in octetstring input, in integer out_length)
return octetstring;
```

The following functions set the keys for `identifier`:

```
function f_set_Ki(in integer identifier, in octetstring input, inout
EAP_port_descriptor descriptor);

function f_set_K(in integer identifier, in octetstring input, inout
EAP_port_descriptor descriptor);

function f_set_SQN(in integer identifier, in octetstring input, inout
EAP_port_descriptor descriptor);

function f_set_SQN_MS(in integer identifier, in octetstring input, inout
EAP_port_descriptor descriptor);

function f_set_AMF(in integer identifier, in octetstring input, inout
EAP_port_descriptor descriptor);
```

The function below calculates `XDOUT`, `Kencr`, `Kaut` and `AK` values. `Kaut` is used when calculating MAC values, `Kencr` is used for encryption and decryption of `AT_ENCR_DATA` attributes, and `AK` is used for calculating and verifying `AT_AUTN` and `AT_AUTS` values.

```
function f_calc_AKA_Keys(in octetstring pl_eap_identity, in octetstring pl_AKA_K,in
octetstring pl_rand, inout octetstring pl_AK,inout octetstring pl_Kaut,inout
octetstring pl_Kencr) return octetstring
```

`A3A8` value is generated from `Ki` key and rand list. It is used in calculating `Kaut`:

```
function f_calc_A3A8(in octetstring key,in octetstring rand)return octetstring;
```

The value `n*SRES` is n SRES values concatenated. It can be generated with the following function from `Ki` key and rand list:

```
function f_calc_SRES(in octetstring key,in octetstring rand)return octetstring;
```

When generating `Kaut` and `Kenc` the input octetstring is concatenated from `identifier`, `A3A8`, `nonce_mt`, version list and selected version.

```
function f_calc_Kaut(in octetstring input,inout octetstring kencr) return octetstring;
```

The next function is used in `f_crypt_atSimEncrData` and `f_crypt_atAKAEncrData`. It generates `AES_cbc_encrypted` or decrypted value. `Kenc` key and `ivec` is needed for calculation.

```
function f_encrypt_at_encr(in octetstring key,in octetstring input,in octetstring
ivec,in boolean decrypt) return octetstring;
```

Functions for encryption or decryption. `Kenc` and `ivec` are needed.

```
function f_crypt_atSimEncrData( in at_sim_encr_data pl_encr_data, in octetstring
key,in octetstring ivec,in boolean decrypt) return at_sim_encr_data;

function f_crypt_atAKAEncrData(in at_aka_encr_data pl_encr_data, in octetstring key,in
octetstring ivec,in boolean decrypt)return at_aka_encr_data;
```

Example

There are no examples available for this protocol module.

Terminology

TITAN TTCN-3 Test Executor.

Abbreviations

CHAP
: PPP Challenge Handshake Authentication Protocol

IETF
: Internet Engineering Task Force

IP
: Internet Protocol

IPCP
: PPP Internet Protocol Control Protocol

LCP
: Link Control Protocol

PAP
: PPP Authentication Protocols

EAP
: Extensible Authentication Protocol

PPP
: Point-to-Point Protocol

RFC
: Request for Comments

TTCN-3

Testing and Test Control Notation version 3

References

[1] ETSI ES 201 873–1 v.3.2.1 (2007-02)
The Testing and Test Control Notation version 3. Part 1: Core Language

[2] Programmer's Technical Reference for the TITAN TTCN-3 Test Executor

[3] Installation Guide for the TITAN TTCN-3 Test Executor

[4] PPP Protocol Modules for TTCN-3 Toolset with TITAN, Function Specification

[5] IETF RFC 1661
The Point-to-Point Protocol

[6] IETF RFC 1332
The PPP Internet Protocol Control Protocol (IPCP)

[7] IETF RFC 1877
PPP Internet Protocol Control Protocol Extensions for Name Server Addresses

[8] IETF RFC 1994
PPP Challenge Handshake Authentication Protocol (CHAP)

[9] IETF RFC 1334
PPP Authentication Protocols

[10] IETF RFC 1662
PPP in HDLC-like Framing

[11] IETF RFC 3748
Extensible Authentication Protocol (EAP)

[12] Extensible Authentication Protocol Method for GSM Subscriber Identity Modules (EAP-SIM)
https://tools.ietf.org/html/draft-haverinen-pppext-eap-sim-16 (2004-12)

[13] Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement + https://tools.ietf.org/html/draft-arkko-pppext-eap-aka-15 (2004-12)

[14] IP Protocol Modules for TTCN-3 Toolset with TITAN, Function Specification