|  |  |
| --- | --- |
|  |  |

EAP Protocol Module for TTCN-3 Toolset with TITAN, Description

# Abstract

This is the description for the EAP protocol modules. The EAP protocol modules are developed for the TTCN-3 Toolset with Titan. This document should be read together with Product Revision Information [8].

Contents

[1 Introduction 2](#_Toc473791833)

[1.1 Revision history 2](#_Toc473791834)

[2 About this Document 2](#_Toc473791835)

[2.1 Presumed Knowledge 2](#_Toc473791836)

[2.2 System Requirements 2](#_Toc473791837)

[3 Usage 2](#_Toc473791838)

[3.1 Overview 2](#_Toc473791839)

[3.2 Installation 3](#_Toc473791840)

[3.3 Configuration 3](#_Toc473791841)

[3.4 Implemented protocols 3](#_Toc473791842)

[3.5 Implemented messages 3](#_Toc473791843)

[3.6 Protocol Modifications/Deviations 3](#_Toc473791844)

[3.7 Backward incompatibilities 3](#_Toc473791845)

[4 Implementation specifics 3](#_Toc473791846)

[5 Interface description 4](#_Toc473791847)

[5.1 Encoding/decoding and other related functions 4](#_Toc473791848)

[5.2 Implemented encoding and decoding functions 4](#_Toc473791849)

[5.3 Authentication and encryption key generation 4](#_Toc473791850)

[6 Examples 6](#_Toc473791851)

[6.1 EAP packet encoding and decoding 6](#_Toc473791852)

[6.2 Computation of MAC parameter in an EAP message 6](#_Toc473791853)

[7 Terminology 6](#_Toc473791854)

[7.1 Abbreviations 6](#_Toc473791855)

[7.2 Terminology 7](#_Toc473791856)

[8 References 7](#_Toc473791857)

# Introduction

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Characteristics | Prepared |
| 2014-02-03 | A | CR\_TR00020239 | EESZSUS |
| 2016-06-20 | B | [artf749725](https://eforge.ericsson.se/sf/go/artf749725) Removed non-working functions | EKLMMIK |
| 2016-07-12 | C | [artf759315](https://eforge.ericsson.se/sf/go/artf759315) Added variable definitions | EKLMMIK |
| 2016-08-05 | D | [artf764108](https://eforge.ericsson.se/sf/go/artf764108?returnUrlKey=1470381151668) Added functions documentations | EKLMMIK |
| 2017-01-27 | E | [734224](https://openalm.lmera.ericsson.se/plugins/tracker/?aid=734224) Added a function for fast re-authentication XKEY’ calculation. | EMATEKO |

# About this Document

## 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 version CRL 113 200 R7A (1.7.pl0) or higher installed. For Installation Guide see [2]. Please note: This version of the test port is not compatible with Titan releases earlier than CRL 113 200 R7A.

# Usage

## Overview

Protocol modules implement the messages structure of the related 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 is usable with the TITAN test toolset only.

## 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 GTPv2 v10.6.0 protocol is used as a part of a TTCN-3 test suite, this requires TTCN-3 Test Executor 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 section of [2].

## Configuration

None.

## Implemented protocols

This set of protocol modules implements protocol messages and constants of RFCs of EAP, EAP-AKA, EAP-AKA’, EAP-SIM, EAP-TTLS.

## Implemented messages

The following messages will be implemented: All message defined in [3], [4], [5], [6] and [7].

## Protocol Modifications/Deviations

There are currently no deviations from the mentioned standards.

## Backward incompatibilities

None.

# Implementation specifics

The enc\_EAP\_PDU and dec\_EAP\_PDU are used to encode and decode the given TTCN PDU\_EAP type to octetsting and vice versa respectedly.

The other functions implemented in the Protocol Module are used in junction with the EAP protocol to compute the various parameters in the EAP messages. A usage example can be found at section 4.2. List of the functions implemented can be found in the Function Specification.

# Interface description

## Encoding/decoding and other related functions

This product also contains encoding/decoding functions that assure correct encoding of messages when sent from Titan and correct decoding of messages when received by Titan. Other implemented functions are used with the EAP protocol parameter computing.

## Implemented encoding and decoding functions

Name Type of formal parameters Type of return value

**enc\_PDU\_EAP PDU\_EAP octetstring  
dec\_PDU\_EAP octetstring PDU\_EAP  
enc\_AKA\_Attrib EAP\_AKA\_Attrib\_List octetstring  
dec\_AKA\_Attrib octetstring EAP\_AKA\_Attrib\_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**

**f\_enc\_tls\_handshakeData TLS\_HandshakeData octetstring**

**f\_dec\_tls\_handshakeData octetstring TLS\_HandshakeData\_t**

**eap\_sim\_derive\_mk octetstring octetstring  
fips186\_2\_prf octetstring octetstring  
eap\_aka\_derive\_mk octetstring octetstring  
eap\_aka\_derive\_reauth\_msk\_emsk octetstring octetstring  
eap\_akaprime\_derive\_mk octetstring octetstring  
Calculate\_AT\_CheckCode octetstring octetstring**

**f\_calc\_Kaut octetstring octetstring**

**f\_calc\_AKA\_Keys octetstring octetstring**

**f\_get\_ServersPublicKey octetstring octetstring**

**f\_prf octetstring octetstring**

## Authentication and encryption key generation

The following functions are implemented in TCCUsefulFunctions\_CNL113472 [9] / TCCSecurity\_Functions module, which were earlier part of the EAP Protocol Module:

Deleted Use this from TCCSecurity\_Functions

A3A8 f\_EAPSIM\_A3A8

hmac\_sha1\_128\_vector f\_calculate\_HMAC\_SHA1

aes\_128\_cbc\_decrypt f\_AES\_CBC\_128\_Decrypt\_OpenSSL

aes\_128\_cbc\_encrypt f\_AES\_CBC\_128\_Encrypt\_OpenSSL

f1 f\_IMSAKA\_f1

f2345 f\_IMSAKA\_f2345

f1star f\_IMSAKA\_f1star

f5star f\_IMSAKA\_f5star

akaprime\_hmac\_sha256\_vector f\_calculate\_HMAC\_SHA256

f\_calc\_SRES f\_EAPSIM\_A3A8

f\_sha1\_256 f\_calculate\_HMAC\_SHA256

f\_sha1 f\_calculate\_HMAC\_SHA1

f\_md5 f\_calculateMD5

f\_calc\_EAPSIM\_Keys f\_EAPSIM\_A3A8

A Master Key is derived from the underlying GSM authentication values (Kc keys), the nonce\_mt, and other relevant context as follows.

function eap\_sim\_derive\_mk(octetstring identity , octetstring nonce\_mt,integer selected\_version, octetstring ver\_list, octetstring kc) return octetstring;

On EAP-AKA full authentication, a Master Key (MK) is derived from the underlying AKA values (CK and IK keys), and the identity, as follows.

function eap\_aka\_derive\_mk(octetstring identity, octetstring ik, octetstring ck) return octetstring;

On EAP-AKA fast re-authentication, the XKEY’ as the seed value of the pseudo-random generator for the Master Session Key (MSK) and the Extended Master Session Key (EMSK) is derived from the underlying AKA value (Master Key), the nonce\_s and the counter values and the identity, as follows.

function eap\_aka\_derive\_reauth\_msk\_emsk(octetstring identity, octetstring counter, octetstring nonce\_s, octetstring mk) return octetstring;

The checkcode is a hash value, calculated with SHA1 [SHA-1], over all EAP-Request/AKA-Identity and EAP-Response/AKA-Identity packets exchanged in the authentication exchange.

function Calculate\_AT\_CheckCode(octetstring rcveap,octetstring sendeap) 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 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

eap\_sim\_derive\_mk rfc4186

eap\_aka\_derive\_mk rfc4187

eap\_aka\_derive\_reauth\_msk\_emsk rfc4187

Calculate\_AT\_CheckCode rfc4187

f\_calc\_Kaut rfc5448

f\_get\_ServersPublicKey rfc5247

f\_prf rfc5281

# Examples

## EAP packet encoding and decoding

The following example shows how an EAP packet can be encoded and decoded.

var PDU\_EAP v\_EAP\_PDU;

var octetstring data;

data:= enc\_PDU\_EAP(v\_EAP\_PDU);

v\_EAP\_PDU := dec\_PDU\_EAP(data);

## Computation of MAC parameter in an EAP message

The following example shows how an EAP packet MAC parameter can be computed with the

import from TCCSecurity\_Functions all;  
[…]

var EAP\_PDU v\_packetToSend\_EAP;

var octetstring v\_ck, v\_ik, v\_name;

var octetstring vl\_mk:= eap\_akaprime\_derive\_mk(v\_name, v\_ik,v\_ck);

var octetstring vl\_k\_aut := substr(vl\_mk,16,32);

var octetstring vl\_macFull :=

f\_calculate\_HMAC\_SHA256 (vl\_k\_aut,enc\_PDU\_EAP(v\_packetToSend\_EAP),32);

var octetstring vl\_mac :=substr(vl\_macFull,0,16);

# Terminology

## Abbreviations

EAP Extensible Authentication Protocol

PDU Protocol Data Unit

IETF Internet Engineering Task Force

TTCN-3 Testing and Test Control Notation version 3

## Terminology

TITAN TTCN-3 Test Executor (see [2]).

# References

1. ETSI ES 201 873-1 v4.5.1 (2013-04)   
   The Testing and Test Control Notation version 3. Part 1: Core Language
2. 1/ 198 17-CRL 113 200/5 Uen   
   User Guide for TITAN TTCN-3 Test Executor
3. IETF RFC 3748 Extensible Authentication Protocol (EAP)
4. IETF RFC 4187 Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)
5. IETF RFC 5448 Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')
6. IETF RFC 4186 Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)
7. IETF RFC 5281 Extensible Authentication Protocol Tunneled Transport Layer Security  
   Authenticated Protocol Version 0 (EAP-TTLSv0)
8. 109 21-CNL 113 722-6 Uen  
   EAP Protocol Modules for TTCN-3 Toolset with TITAN, Product Revision Information
9. 1551-CNL 113 427 Uen  
   TCC Useful Functions for TTCN-3 Toolset with TITAN