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

# Abstract

The purpose of this document is to specify the functionality and usage of the TLS protocol module.

Contents

[1 Functionality 2](#_Toc475966045)

[1.1 Protocol version implemented 2](#_Toc475966046)

[1.2 Modifications/deviations related to the protocol specification 2](#_Toc475966047)

[1.2.1 Unimplemented Messages, Information Elements and Constants 2](#_Toc475966048)

[1.2.2 Protocol Modifications/Deviations 2](#_Toc475966049)

[1.3 System Requirements 3](#_Toc475966050)

[1.4 Installation 3](#_Toc475966051)

[1.5 Examples 3](#_Toc475966052)

[1.5.1 Example for message length calculation and message splitting 3](#_Toc475966053)

[1.6 Encoder, decoder functions 4](#_Toc475966054)

[1.6.1 Implemented encoding/decoding functions: 4](#_Toc475966055)

[2 Abbreviations 4](#_Toc475966056)

[3 References 5](#_Toc475966057)

[4 Change information 5](#_Toc475966058)

[4.1 R1A 5](#_Toc475966059)

# Introduction

This is the Function Description for the set TLS protocol modules. CAPv1 protocol modules are developed for the TTCN-3 Toolset with TITAN. This document should be read together with Product Revision Information [2].

# Functionality

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

## Protocol version implemented

This set of protocol modules implements protocol messages and constants of the TLS protocol, v1.1 (see [4]) with the modifications specified in 1.2.)

Implemented extensions:

* Server Name Indication (SNI) in the Client Hello (RFC 6066)
* Application Layer Protocol (RFC 7301)

## Modifications/deviations related to the protocol specification

The standard TLS protocol modules had to be changed in order to compile and become useable. These changes are detailed below.

### Unimplemented Messages, Information Elements and Constants

None

### Protocol Modifications/Deviations

Protocol modules contain the following modifications/deviations from [4] changing the protocol message structure/behaviour.

#### If a Handshake message decoding fails it will be an octetstring with TLSText.length bytes and decoded as Encrypted Handshake.

#### For the opportunity to extend the ClientHello message, new field named extension\_ added.

#### KeyExchangeAlgorithm named null\_ added, to use as default value in some decoding function.

#### This protocol module implementation does not support to send encrypted messages, only support handshakes, alerts and cipher spec changes.

## System Requirements

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 R7A or higher installed. For installation guide see [3]. Please note: This version of the protocol module is not compatible with TITAN releases earlier than R7A.

## 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 (e.g. nedit, xemacs). Since the TLS 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 [3].

## Examples

### Example for message length calculation and message splitting

To access this functionality, you have to use the IPL4 Testport’s [5] function called f\_IPL4\_fixedMsgLen. It has two parameters. The first is the message in octetstring (stream), the second is a record of integers which has to be the following:

var ro\_integer args := { 3, 2, 0, 1, 0 }

Now your function call should looks like this:

var integer vl\_len := f\_IPL4\_fixedMsgLen(stream, args)

And now you can easily get one TLS message from the stream with the built-in substr function.

var octetstring vl\_message := substr(stream, 0, len)

## Encoder, decoder functions

### Implemented encoding/decoding functions:

|  |  |  |
| --- | --- | --- |
| **Name** | **Type of formal parameters** | **Type of return value** |
| enc\_TLS\_TLSPlaintexts | TLSPlaintext | octetstring |
| dec\_TLS\_TLSPlaintexts | octetstring, TLSPlaintexts, KeyExchangeAlgorithm (optional) | integer |
| enc\_TLS\_TLSCompressed | TLSCompressed | octetstring |
| dec\_TLS\_TLSCompressed | octetstring, TLSCompressed, KeyExchangeAlgorithm (optional) | integer |
| enc\_TLS\_SecurityParameters | SecurityParameters | octetstring |
| dec\_TLS\_SecurityParameters | Octetstring, SecurityParameters | integer |
| enc\_TLS\_TLSCiphertext | TLSCiphertext, TLSCompressed, SecurityParameters | octetstring |
| dec\_TLS\_TLSCiphertext | octetstring, TLSCiphertext, TLSCompressed, SecurityParameters, KeyExchangeAlgorithm | integer |
| enc\_TLS\_Extensions | TLS\_Extensions | octetstring |
| dec\_TLS\_Extensions | octetstring | TLS\_Extensions |

# Abbreviations

ES ETSI Standard

ETSI European Telecommunications Standards Institute

IETF Internet Engineering Task Force

RFC Request for Comments

TLS Transport Layer Security

TTCN-3 Testing and Test Control Notation version 3

# References

1. ETSI ES 201 873-1 v4.5.1  
   The Testing and Test Control Notation version 3; Part 1: Core Language
2. 109 21-CNL 113 806-4 Uen  
   TLS Protocol Modules for TTCN-3 Toolset with TITAN, Product Revision Information
3. 1/198 17-CRL 113 200/6 Uen  
   TITAN User Guide
4. IETF - RFC 4346,   
   The Transport Layer Security (TLS) Protocol Version 1.1
5. 1551 – CNL 113 531 Uen  
   IPL4asp Test Port for TTCN-3 Toolset with TITAN, Description

# Change information

## R1A

Initial release