SRTP Protocol Modules for TTCN-3 Toolset with TITAN DESCRIPTION

Gábor Szalai

Version 1551-CNL 113 769, Rev. B, 2014-02-13

Table of Contents

Abstract

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

Introduction

This is the Function Specification for the set of SRTP protocol modules. SRTP 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.

Protocol Version Implemented

The protocol module is based on the RFC 3711 [4] and RFC 4774 [5]. All mandatory-to-support and optional elements are implemented, with the default values defined by the standards.

Protocol Modifications/Deviations

None

SRTP Cryptographic Context

The protocol module provides data structure to handle the SRTP cryptographic context parameters, described in the 3.2 of RFC 3711 [4].

Context Initialization

Before the context is used it should be initialised by setting the fields of it to the user supplied or default value. There are two templates defined by the protocol module for initialization:

- Context initialization template: `t_SRTP_init_crypto_context`
- Master key initialization template: `t_SRTP_init_master_key`

`t_SRTP_init_crypto_context`

The definition of the template:

```
template SRTP_crypto_context_params
                        t_SRTP_init_crypto_context(
   SRTP_crypto_transform  vl_crypto_param,
   SRTP_auth_transform    vl_auth_param,
   SRTP_master_key_list   vl_master_key_list
  )
```

The template initializes the SRTP context with a given parameters.

`t_SRTP_init_master_key`

The definition of the template:

```
template SRTP_master_key_params t_SRTP_init_master_key(
    octetstring           vl_master_key,
    template octetstring  vl_master_salt:= omit,
    integer               vl_key_derivation_rate:=0
  )
```

The template initializes the master key parameters with a given keys. If the master salt or key derivation rate is not given the default value of them is used.

Example Initialization

```
var SRTP_crypto_context_params context
var SRTP_master_key_params mkey

mkey:=valueof(t_SRTP_init_master_key(
              '5842d9e864c2ca0477ee41795ab32ca2'O,
              'b6f8157eade43a3baf3df9c90c36'O));
context:=valueof(t_SRTP_init_crypto_context(
  {aes_cm:={omit,omit}},{hmac_sha1:={omit,omit}},{mkey}))
```

Session Key Handling

The session keys are automatically generated by the encoder/decoder function using the method described in 4.3 RFC 3711 [4]. The generated session keys are stored in appropriate master key item of the SRTP context. If the session keys are set by the user, the user supplied values will be used.

If more than one key specified the encoder/decoder function choose the correct one using the MKI and/or `<from, to>` values. If no MKI of `<from, to>` ranges are specified, the selection is controlled by the key_index field of the context.

If the key is automatically selected the key_index field is updated with the index of the used key. There are counters for every key, which are automatically updated by the encoder fucntions:

- processed_packets_srtp
- processed_packets_srtcp

Context Selection

The cryptographic context determination is solely the responsibility of the user. There is a predefined field of the context data type to hold the context identifiers, but that field is not used by the protocol module. The user should supply the correct context to the encoder/decoder functions.

Debug Functionality

The user can activate the debug logging by set the tsp_SRTP_debug_log_enabled module parameter to true. Also the DEBUG logging category should be added to the FileMask in the configuration file.

Example:

```
[LOGGING]
FileMask:=LOG_ALL | DEBUG

[MODULE_PARAMETERS]
tsp_SRTP_debug_log_enabled:=true
```

Encoding/Decoding and Other Related Functions

This product also contains encoding/decoding [4] functions which assure correct encoding of messages when sent from Titan and correct decoding of messages when received by TITAN. Implemented encoding/decoding functions:

```
  external function f_SRTP_encoder(
    in RTP_messages_union pl_pdu,
    inout  SRTP_crypto_context_params pl_context,
    out octetstring pl_packet
    );

  external function f_SRTP_decoder(
    in octetstring pl_packet,
    inout  SRTP_crypto_context_params pl_context,
    out RTP_messages_union pl_pdu
    ) return SRTP_result;
```

Terminology

No specific terminology is used.

Abbreviations

TTCN-3	Testing and Test Control Notation version 3
SRTP	Secure Real-time Transport Protocol

References

[1] ETSI ES 201 873-1 V.4.4.1 (2012-04) The Testing and Test Control Notation version 3 Part 1: Core Language

[2] 109 21-CNL 113 769/1-1 Uen SRTP Protocol Modules for TTCN-3 Toolset with TITAN, Product Revision Information

[3] 1/198 17-CRL 113 200/4 Uen - User Guide for TITAN TTCN–3 Test Executor

[4] The Secure Real-time Transport Protocol (SRTP) IETF RFC 3711

[5] Integrity Transform Carrying Roll-Over Counter for the Secure Real-time Transport Protocol IETF RFC 4771