Datagram Transport Layer Security (DTLS) Security Requirements and Evaluation Activities

Contents

[Overview 3](#_Toc456010682)

[Differences from TLS 3](#_Toc456010683)

[TOE Security Functional Requirements 6](#_Toc456010684)

[DTLS Client and DTLS Server Protocol Requirements 6](#_Toc456010685)

[FCS\_DTLSC\_EXT.2 DTLS Client Protocol with authentication 8](#_Toc456010686)

[FCS\_DTLSS\_EXT.1 DTLS Server Protocol 11](#_Toc456010687)

[FCS\_DTLSS\_EXT.2 DTLS Server Protocol with mutual authentication 13](#_Toc456010688)

[Audit Events for DTLS SFRs 16](#_Toc456010689)

[Evaluation Activities for SFRs 17](#_Toc456010690)

[FCS\_DTLSC\_EXT.1 Extended: DTLS Client Protocol 17](#_Toc456010691)

[FCS\_DTLSC\_EXT.2 Extended: DTLS Client Protocol with authentication 20](#_Toc456010692)

[FCS\_DTLSS\_EXT.1 Extended: DTLS Server Protocol 25](#_Toc456010693)

[FCS\_DTLSS\_EXT.2 Extended: DTLS Server Protocol with mutual authentication 28](#_Toc456010694)

[Extended Component Definitions 32](#_Toc456010695)

[FCS\_DTLSC\_EXT DTLS Client Protocol 32](#_Toc456010696)

[FCS\_DTLSS\_EXT DTLS Server Protocol 36](#_Toc456010697)

# Overview

This document contains a set of Security Functional Requirements (SFRs) and Evaluation Activities for Datagram Transport Layer Security (DTLS).

The TLS specification was designed to work over TCP, which provides a connection setup as well as an ordered and error-checked transport of encrypted data. This results in a reliable trusted communication channel. As such, TLS over TCP would not tolerate traffic that was lost or arrived in the wrong order.

DTLS is a modification of the TLS specification to allow for the encrypted transfer of data using a trusted communication channel over the UDP transport. The UDP transport is not connection-oriented, meaning traffic may be lost, duplicated, delayed, or received in the wrong order.

Products implement DTLS for time sensitive application protocols that demand a connectionless transmission model.

Use cases where an international Technical Community (iTC) should consider DTLS are listed below:

* Transport of syslog messages over DTLS as described in RFC 6012
* Transport of SNMP over DTLS as described in RFC 6353
* Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification defined in RFC 5415 required for ND CPP EPs Wireless LAN Access System (published) and Wireless Intrusion Detection/Prevention Systems (draft)
* Transport of RADIUS over DTLS as described in RFC 7360

This document does not cover:

* Use case(s) where a transport protocol other than UDP is implemented.

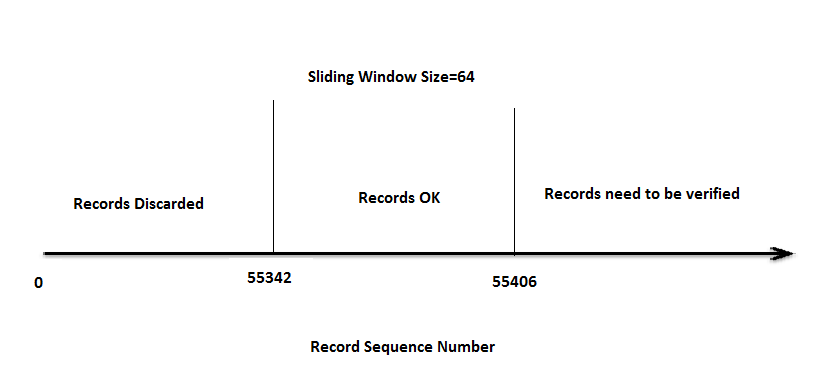
* Use case(s) where DTLS would be an option to secure the connection in FTP\_TRP.1/Admin.

# Differences from TLS

As stated in the section above, TLS cannot be used to secure traffic over an unreliable transport as it requires TCP to provide a connection setup and reliable transport. DTLS as defined in RFC 6347 and RFC 4347 is purposely similar to TLS with provisions to allow a trusted communication channel to transport encrypted data over UDP.

As a background it’s helpful to describe the DTLS differences from TLS:

1. **Explicit sequence number field**. In TLS, the sequence number is in the Message Authentication Code (MAC) which is verified by the receiver only after decryption. This is not possible with UDP transport since records may be lost, duplicated, or arrive out of sequence. DTLS therefore includes a record sequence number to allow the receiver to reorder records correctly.
2. **HelloVerifyRequest.** Since there is no connection setup with UDP as there is in TCP, an attacker could perform a DoS attack by continuously sending ClientHello messages to the server. To prevent this type of DoS attack, an additional handshake message, HelloVerifyRequest, is added. The HelloVerifyRequest along with a stateless cookie exchange emulates the TCP three-way handshake and ensures the DTLS Client is not using a spoofed IP Address.
3. **MAC verification failure**. In TLS, MAC verification failures result in connection termination. DTLS implementations may either silently discard data with bad or invalid MAC and continue with the session or terminate the connection.
4. **Replay Detection.** DTLS does not have a connection setup and ordered/error-checked transport like TLS. This makes DTLS susceptible to replay attacks. The DTLS Server and optionally the DTLS Client shall enforce replay detection using sequence numbers and sliding window protocol. Valid Record Sequence Numbers of received messages are maintained in a sliding window. For example:



For each record received, the receiver shall verify if the record is within the window boundary. Messages that are received where the same record was previously received or is too old to fit in the window shall be silently discarded.

1. **No Stream Ciphers**. Support for stream ciphers (i.e. RC4) is not suitable for DTLS. This is because the decryption of data in stream cipher depends on the previous decrypted record, which in UDP could be lost. This does not present an issue as all TLS ciphersuites in the NDcPP are block ciphers.
2. **No verified connection termination**. TLS has a verified end-of-connection termination. Due to the unreliable nature of UDP, in DTLS, when a receiver stops receiving datagrams from a peer, it doesn’t know whether the sender has voluntarily ceased to send, or whether the rest of the data was lost. To ensure audit requirements are properly met, a DTLS receiver may need to monitor the connection at the application layer. When no data is received from a DTLS connection for a long time (where the application decides what "long" means), the receiver should send a TLS/DTLS close\_notify alert message and close the connection.
3. **Epochs**. In addition to the explicit sequence number field, DTLS also has a field called "epoch". Epochs are a numerical counter used to determine which cipher state has been used to protect the record payload. This allows recipients to distinguish such packets if data loss occurs during a session renegotiation. The epoch is incremented after each ChangeCipherSpec message is sent. In TLS, the transport is reliable so there is no need for this counter.
4. **Retransmission Timer.** In TLS, the TCP transport handles retransmission to avoid packet loss. In DTLS this is handled with a retransmission timer which will retransmit a message after some timeout until a reply is received.

# TOE Security Functional Requirements

The set of TLS requirements in the NDcPP are modified in the section below to apply to DTLS.

## DTLS Client and DTLS Server Protocol Requirements

Datagram TLS (DTLS) is not a required component of the NDcPP. If a TOE implements DTLS, a corresponding selection in FTP\_ITC.1 or FPT\_ITT.1 should be made to define what the DTLS protocol is implemented to protect.

A TOE may act as the client, the server, or both in DTLS sessions. The requirement has been separated into DTLS Client (FCS\_DTLSC\_EXT) and DTLS Server (FCS\_DTLSS\_EXT) requirements to allow for these differences.

If the TOE acts as the client during the claimed DTLS sessions, the ST author should claim one of the FCS\_DTLSC\_EXT requirements. If the TOE only transmits application-layer data to an external entity using a trusted channel provided by DTLS, (i.e. transmits syslog over DTLS) then FCS\_DTLSC\_EXT.1 should be selected.

If the application layer communication is bi-directional, that is, the TOE both transmits and receives application data or is managed by the DTLS Server, then FCS\_DTLSC\_EXT.2 is required. FCS\_DTLSC\_EXT.2 requires the client must be capable of the following:

* Present a certificate to a DTLS Server for mutual authentication.
* Perform a selected action if a DTLS message from the DTLS Server contains an invalid Message Authentication Code (MAC).
* Detect replayed messages

To ensure audit requirements are properly met a DTLS receiver may need to monitor the DTLS connection state at the application layer. When no data is received from a DTLS connection for a long time (where the application decides what "long" means), the receiver should send a close\_notify alert message and close the connection.

**FCS\_DTLSC\_EXT.1 DTLS Client Protocol**

**FCS\_DTLSC\_EXT.1.1** The TSF shall implement [selection: *DTLS 1.2 (RFC 6347), DTLS 1.0 (RFC 4347)*] supporting the following ciphersuites:

* [selection:
  + *TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5246*
  + *TLS\_RSA\_WITH\_AES\_256\_CBC\_ SHA256 as defined in RFC 5246*
  + *TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_ SHA256 as defined in RFC 5246*
  + *TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_ SHA256 as defined in RFC 5246*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5289*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384 as defined in RFC 5289*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5289*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5289*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5289*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5289*

].

The ciphersuites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional ciphersuites that are supported. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA is required in order to ensure compliance with RFC 5246.

These requirements will be revisited as new DTLS versions are standardized by the IETF.

In a future version of this cPP DTLS v1.2 will be required for all TOEs.

**FCS\_DTLSC\_EXT.1.2** The TSF shall verify that the presented identifier matches the reference identifier according to RFC 6125 section 6.

The rules for verification of identify are described in Section 6 of RFC 6125. The reference identifier is established by the user (e.g. entering a URL into a web browser or clicking a link), by configuration (e.g. configuring the name of a mail server or authentication server), or by an application (e.g. a parameter of an API) depending on the application service. Based on a singular reference identifier’s source domain and application service type (e.g. HTTP, SIP, LDAP), the client establishes all reference identifiers which are acceptable, such as a Common Name for the Subject Name field of the certificate and a (case-insensitive) DNS name, URI name, and Service Name for the Subject Alternative Name field. The client then compares this list of all acceptable reference identifiers to the presented identifiers in the DTLS server’s certificate.

The preferred method for verification is the Subject Alternative Name using DNS names, URI names, or Service Names. Verification using the Common Name is required for the purposes of backwards compatibility. Additionally, support for use of IP addresses in the Subject Name or Subject Alternative name is discouraged as against best practices but may be implemented. Finally, the client should avoid constructing reference identifiers using wildcards. However, if the presented identifiers include wildcards, the client must follow the best practices regarding matching; these best practices are captured in the assurance activity.

**FCS\_DTLSC\_EXT.1.3** The TSF shall only establish a trusted channel if the peer certificate is valid.

If DTLS is selected in FTP\_ITC then validity is determined by the identifier verification, certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity is tested in accordance with testing performed for FIA\_X509\_EXT.1/Rev. If DTLS is selected in FPT\_ITT then certificate validity is tested in accordance with testing performed for FIA\_X509\_EXT.1/ITT.

**FCS\_DTLSC\_EXT.1.4** The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [selection: *secp256r1, secp384r1, secp521r1, or none*] and no other curves.

If ciphersuites with elliptic curves were selected in FCS\_DTLSC\_EXT.1.1, a selection of one or more curves is required. If no ciphersuites with elliptic curves were selected in FCS\_DTLS\_EXT.1.1, then ‘none’ should be selected.

This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS\_COP.1//SigGen and FCS\_CKM.1 and FCS\_CKM.2. This extension is required for clients supporting Elliptic Curve ciphersuites.

## FCS\_DTLSC\_EXT.2 DTLS Client Protocol with authentication

**FCS\_DTLSC\_EXT.2 DTLS Client Protocol – with authentication**

**FCS\_DTLSC\_EXT.2.1** The TSF shall implement [selection: *DTLS 1.2 (RFC 6347), DTLS 1.0 (RFC 4347)*] supporting the following ciphersuites:

* [selection:
  + *TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5246*
  + *TLS\_RSA\_WITH\_AES\_256\_CBC\_ SHA256 as defined in RFC 5246*
  + *TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_ SHA256 as defined in RFC 5246*
  + *TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_ SHA256 as defined in RFC 5246*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5289*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384 as defined in RFC 5289*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5289*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5289*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5289*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5289*

].

The ST author should select the optional ciphersuites that are supported. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. The Suite B algorithms listed above (RFC 6460) are the preferred algorithms for implementation. TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA is required in order to ensure compliance with RFC 5246.

These requirements will be revisited as new DTLS versions are standardized by the IETF.

In a future version of this cPP DTLS v1.2 will be required for all TOEs.

**FCS\_DTLSC\_EXT.2.2** The TSF shall verify that the presented identifier matches the reference identifier according to RFC 6125 section 6.

The rules for verification of identify are described in Section 6 of RFC 6125. The reference identifier is established by the user (e.g. entering a URL into a web browser or clicking a link), by configuration (e.g. configuring the name of a mail server or authentication server), or by an application (e.g. a parameter of an API) depending on the application service. Based on a singular reference identifier’s source domain and application service type (e.g. HTTP, SIP, LDAP), the client establishes all reference identifiers which are acceptable, such as a Common Name for the Subject Name field of the certificate and a (case-insensitive) DNS name, URI name, and Service Name for the Subject Alternative Name field. The client then compares this list of all acceptable reference identifiers to the presented identifiers in the DTLS server’s certificate.

The preferred method for verification is the Subject Alternative Name using DNS names, URI names, or Service Names. Verification using the Common Name is required for the purposes of backwards compatibility. Additionally, support for use of IP addresses in the Subject Name or Subject Alternative name is discouraged as against best practices but may be implemented. Finally, the client should avoid constructing reference identifiers using wildcards. However, if the presented identifiers include wildcards, the client must follow the best practices regarding matching; these best practices are captured in the assurance activity.

**FCS\_DTLSC\_EXT.2.3** The TSF shall only establish a trusted channel if the peer certificate is valid.

If DTLS is selected in FTP\_ITC then validity is determined by the identifier verification, certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity is tested in accordance with testing performed for FIA\_X509\_EXT.1/Rev. If DTLS is selected in FPT\_ITT then certificate validity is tested in accordance with testing performed for FIA\_X509\_EXT.1/ITT.

**FCS\_DTLSC\_EXT.2.4** The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [selection: secp256r1, secp384r1, secp521r1, or none] and no other curves.

If ciphersuites with elliptic curves were selected in FCS\_DTLSC\_EXT.2.1, a selection of one or more curves is required. If no ciphersuites with elliptic curves were selected in FCS\_DTLS\_EXT.2.1, then ‘none’ should be selected.

*This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS\_COP.1/SigGen and FCS\_CKM.1 and FCS\_CKM.2. This extension is required for clients supporting Elliptic Curve ciphersuites.*

**FCS\_DTLSC\_EXT.2.5** The TSF shall support mutual authentication using X.509v3 certificates.

The use of X.509v3 certificates for DTLS is addressed in FIA\_X509\_EXT.2.1. This requirement adds that the client must be capable of presenting a certificate to a DTLS server for DTLS mutual authentication.

**FCS\_DTLSC\_EXT.2.6** The TSF shall [selection: *terminate the DTLS session, silently discard the record*] if a message received contains an invalid MAC.

*The Message Authentication Code (MAC) is keyed hash function specified in FCS\_COP.1(4). The MAC is negotiated during DTLS handshake phase and is used to protect integrity of messages received from the sender during DTLS data exchange. If MAC verification fails the session must be terminated or the record must be silently discarded.*

**FCS\_DTLSC\_EXT.2.7** The TSF shall detect and silently discard replayed messages for:

* DTLS records previously received.
* DTLS records too old to fit in the sliding window.

Replay Detection is described in section 4.1.2.6 of DTLS 1.2 (RFC 6347) and section 4.1.2.5 of DTLS 1.1 (RFC 4347). For each received record, the receiver verifies the record contains a sequence number that is within the sliding receive window and does not duplicate the sequence number of any other record received during the session.

*"*Silently *Discard" means the TOE discards the packet without further processing*

## FCS\_DTLSS\_EXT.1 DTLS Server Protocol

As discussed in the introductory section the TOE may act as the client, the server, or both in DTLS sessions. If the TOE acts as the server during the claimed DTLS sessions (FTP\_ITC.1 or FPT\_ITT.1), the ST author should claim one of the FCS\_DTLSS\_EXT claims.

DTLS may or may not be performed with mutual authentication. The ST author shall claim FCS\_DTLSS\_EXT.1 if the TOE does not support mutual authentication. The ST author should claim FCS\_DTLSS\_EXT.2 if mutual authentication is supported by the TOE.

FCS\_DTLSS\_EXT.1 and FCS\_DTLSS\_EXT.2 requires the client must be capable of the following:

* Perform a selected action if a DTLS message from the DTLS Server contains an invalid Message Authentication Code (MAC).
* Detect replayed messages.
* Perform validation of the DTLS Client during connection handshake.

**FCS\_DTLSS\_EXT.1 DTLS Server Protocol**

**FCS\_DTLSS\_EXT.1.1** The TSF shall implement [selection: *DTLS 1.2 (RFC 6347), DTLS 1.0 (RFC 4347)*] supporting the following ciphersuites:

* [selection:
  + *TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5246*
  + *TLS\_RSA\_WITH\_AES\_256\_CBC\_ SHA256 as defined in RFC 5246*
  + *TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_ SHA256 as defined in RFC 5246*
  + *TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_ SHA256 as defined in RFC 5246*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5289*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384 as defined in RFC 5289*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5289*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5289*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5289*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5289*

].

The ciphersuites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional ciphersuites that are supported. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA is required in order to ensure compliance with RFC 5246.

These requirements will be revisited as new DTLS versions are standardized by the IETF.

In a future version of this cPP DTLS v1.2 will be required for all TOEs.

**FCS\_DTLSS\_EXT.1.2** The TSF shall deny connections from clients requesting [*none*].

In a future version of this cPP DTLS v1.0 will be required to be denied for all TOEs.

**FCS\_DTLSS\_EXT.1.3** The TSF shall not proceed with a connection handshake attempt if the DTLS Client fails validation.

*The process to validate the DTLS client is specified in section 4.2.1 of RFC 6347 (DTLS 1.2) and RFC 4347 (DTLS 1.1). The TOE validates the DTLS client during Connection Establishment (Handshaking) and prior to the TSF sending a Server Hello message. After receiving a ClientHello, the DTLS Server sends a HelloVerifyRequest along with a cookie. The cookie is a signed message using the keyed hash function specified in FCS\_COP.1(4). The DTLS Client then sends another ClientHello with the cookie attached. If the DTLS server successfully verifies the signed cookie, the Client is not using a spoofed IP address.*

**FCS\_DTLSS\_EXT.1.4** The TSF shall generate key establishment parameters using RSA with key size [selection: *2048 bits, 3072 bits, 4096 bits, no other size*] and [selection: *over NIST curves* [selection: *secp256r1, secp384r1, secp521r1] and no other curves; Diffie-Hellman parameters of size 2048 bits and* [selection: *3072 bits, no other size*]*; no other*].

If the ST lists a DHE or ECDHE ciphersuite in FCS\_DTLSS\_EXT.1.1, the ST must include the Diffie-Hellman or NIST curves selection in the requirement. FMT\_SMF.1 requires the configuration of the key agreement parameters in order to establish the security strength of the DTLS connection.

**FCS\_DTLSS\_EXT.1.5** The TSF shall [selection: *terminate the DTLS session, silently discard the record*] if a message received contains an invalid MAC.

*The Message Authentication Code (MAC) is keyed hash function specified in FCS\_COP.1(4). The MAC is negotiated during DTLS handshake phase and is used to protect integrity of messages received from the sender during DTLS data exchange. If MAC verification fails the session must be terminated or the record must be silently discarded.*

**FCS\_DTLSS\_EXT.1.6** The TSF shall detect and silently discard replayed messages for:

* DTLS records previously received.
* DTLS records too old to fit in the sliding window.

Replay Detection is described in section 4.1.2.6 of DTLS 1.2 (RFC 6347) and section 4.1.2.5 of DTLS 1.1 (RFC 4347). For each received record, the receiver verifies the record contains a sequence number that is within the sliding receive window and does not duplicate the sequence number of any other record received during the session.

"Silently Discard" means the TOE discards the packet without further processing.

## 

## FCS\_DTLSS\_EXT.2 DTLS Server Protocol with mutual authentication

**FCS\_DTLSS\_EXT.2 DTLS Server Protocol with mutual authentication**

**FCS\_DTLSS\_EXT.2.1** The TSF shall implement [selection: *DTLS 1.2 (RFC 6347), DTLS 1.0 (RFC 4347)*] supporting the following ciphersuites:

* [selection:
  + *TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 3268*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 4492*
  + *TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5246*
  + *TLS\_RSA\_WITH\_AES\_256\_CBC\_ SHA256 as defined in RFC 5246*
  + *TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_ SHA256 as defined in RFC 5246*
  + *TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_ SHA256 as defined in RFC 5246*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5289*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384 as defined in RFC 5289*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5289*
  + *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5289*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5289*
  + *TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5289*

].

The ciphersuites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional ciphersuites that are supported. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. The Suite B algorithms listed above (RFC 6460) are the preferred algorithms for implementation. TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA is required in order to ensure compliance with RFC 5246.

These requirements will be revisited as new DTLS versions are standardized by the IETF.

In a future version of this cPP DTLS v1.2 will be required for all TOEs.

**FCS\_DTLSS\_EXT.2.2** The TSF shall deny connections from clients requesting [*none*].

In a future version of this cPP DTLS v1.0 will be required to be denied for all TOEs.

**FCS\_DTLSS\_EXT.2.3** The TSF shall not proceed with a connection handshake attempt if the DTLS Client fails validation.

*The process to validate the DTLS client is specified in section 4.2.1 of RFC 6347 (DTLS 1.2) and RFC 4347 (DTLS 1.1). The TOE validates the DTLS client during Connection Establishment (Handshaking) and prior to the TSF sending a Server Hello message. After receiving a ClientHello, the DTLS Server sends a HelloVerifyRequest along with a cookie. The cookie is a signed message using the keyed hash function specified in FCS\_COP.1(4). The DTLS Client then sends another ClientHello with the cookie attached. If the DTLS server successfully verifies the signed cookie, the Client is not using a spoofed IP address.*

**FCS\_DTLSS\_EXT.2.4** The TSF shall generate key establishment parameters using RSA with key size [selection: *2048 bits, 3072 bits, 4096 bits, no other size*] and [selection: *over NIST curves* [selection: *secp256r1, secp384r1] and no other curves; Diffie-Hellman parameters of size 2048 bits and* [selection: *3072 bits, no other size*]*; no other*].

If the ST lists a DHE or ECDHE ciphersuite in FCS\_TLSS\_EXT.2.1, the ST must include the Diffie-Hellman or NIST curves selection in the requirement. FMT\_SMF.1 requires the configuration of the key agreement parameters in order to establish the security strength of the DTLS connection.

**FCS\_DTLSS\_EXT.2.5** The TSF shall [selection: *terminate the DTLS connection, silently discard the record*] if a message received contains an invalid MAC.

*The Message Authentication Code (MAC) is negotiated during the DTLS handshake phase and is used to protect integrity of messages received from the sender during DTLS data exchange. If MAC verification fails the session must be terminated or silently discarded.*

**FCS\_DTLSS\_EXT.2.6**The TSF shall detect and silently discard replayed messages for:

* DTLS records previously received.
* DTLS records too old to fit in the sliding window.

Replay Detection is described in section 4.1.2.6 of DTLS 1.2 (RFC 6347) and section 4.1.2.5 of DTLS 1.1 (RFC 4347). For each received record, the receiver verifies the record contains a sequence number that is within the sliding receive window and does not duplicate the sequence number of any other record received during the session.

*"Silently* *Discard" means the TOE discards the packet without further processing.*

**FCS\_DTLSS\_EXT.2.7** The TSF shall support mutual authentication of DTLS clients using X.509v3 certificates.

**FCS\_DTLSS\_EXT.2.8** The TSF shall not establish a trusted channel if the peer certificate is invalid.

The use of X.509v3 certificates for DTLS is addressed in FIA\_X509\_EXT.2.1. This requirement adds that this use must include support for client-side certificates for DTLS mutual authentication.

If DTLS is selected in FTP\_ITC then validity is determined by the identifier verification, certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity is tested in accordance with testing performed for FIA\_X509\_EXT.1/Rev. If DTLS is selected in FPT\_ITT then certificate validity is tested in accordance with testing performed for FIA\_X509\_EXT.1/ITT.

**FCS\_DTLSS\_EXT.2.9** The TSF shall not establish a trusted channel if the distinguished name (DN) or Subject Alternative Name (SAN) contained in a certificate does not match the expected identifier for the peer.

The peer identifier may be in the Subject field or the Subject Alternative Name extension of the certificate. The expected identifier may either be configured, may be compared to the Domain Name, IP address, username, or email address used by the peer, or may be passed to a directory server for comparison. Matching should be performed by a bit-wise comparison.

# Audit Events for DTLS SFRs

|  |  |  |
| --- | --- | --- |
| Requirement | Auditable Events | Additional Audit Record Contents |
| FCS\_DTLSC\_EXT.1 | Failure to establish a DTLS session | Reason for failure |
| FCS\_DTLSC\_EXT.2 | Failure to establish a DTLS session | Reason for failure |
| FCS\_DTLSC\_EXT.2 | Detected replay attacks | Identity (e.g., source IP address) of the source of the replay attack. |
| FCS\_DTLSS\_EXT.1 | Failure to establish a DTLS session | Reason for failure |
| FCS\_DTLSS\_EXT.1 | Detected replay attacks | Identity (e.g., source IP address) of the source of the replay attack. |
| FCS\_DTLSS\_EXT.2 | Failure to establish a DTLS session | Reason for failure |
| FCS\_DTLSS\_EXT.2 | Detected replay attacks | Identity (e.g., source IP address) of the source of the replay attack. |

Table 1: DTLS SFRs and Auditable Events

### 

# Evaluation Activities for SFRs

### FCS\_DTLSC\_EXT.1 Extended: DTLS Client Protocol

#### TSS

FCS\_DTLSC\_EXT.1.1

1. The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component.

FCS\_DTLSC\_EXT.1.2

1. The evaluator shall ensure that the TSS describes the client’s method of establishing all reference identifiers from the administrator/application-configured reference identifier, including which types of reference identifiers are supported (e.g Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the TOE.

FCS\_DTLSC\_EXT.1.4

1. The evaluator shall verify that TSS describes the Supported Elliptic Curves Extension and whether the required behaviour is performed by default or may be configured

#### Guidance Documentation

FCS\_DTLSC\_EXT.1.1

1. The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that DTLS conforms to the description in the TSS.

FCS\_DTLSC\_EXT.1.2

1. The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in DTLS.

FCS\_DTLSC\_EXT.1.4

1. If the TSS indicates that the Supported Elliptic Curves Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves Extension.

#### Tests

FCS\_DTLSC\_EXT.1.1

1. Test 1: The evaluator shall establish a DTLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level application protocol, e.g., as part of a syslog session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).
2. Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.
3. Test 3: The evaluator shall send a server certificate in the DTLS connection that does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA ciphersuite). The evaluator shall verify that the TOE disconnects after receiving the server’s Certificate handshake message.
4. Test 4: The evaluator shall configure the server to select the TLS\_NULL\_WITH\_NULL\_NULL ciphersuite and verify that the client denies the connection. Test 2 in FCS\_DTLSS\_EXT.1.1 or FCS\_DTLSS\_EXT.2.1 can be used as a substitute for this test.
5. Test 5: The evaluator perform the following modifications to the traffic:
6. Change the DTLS version selected by the server in the Server Hello to a non-supported DTLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.
7. Modify at least one byte in the server’s nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client’s Finished handshake message.
8. Modify the server’s selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.
9. Modify the signature block in the Server’s Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.
10. Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.
11. Send a garbled message from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.

FCS\_DTLSC\_EXT.1.2

1. The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a DTLS connection:
2. Test 1: The evaluator shall present a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. The evaluator shall verify that the connection fails.
3. Test 2: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.
4. Test 3: The evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds.
5. Test 4: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.
6. Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier:
7. The evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.\*.example.com) and verify that the connection fails.
8. The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. \*.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.come) and verify that the connection fails.
9. Test 6: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.
10. Test 7: [conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

FCS\_DTLSC\_EXT.1.3

1. Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. If the certificate is validated and a trusted channel is established, the test passes. The evaluator then shall delete one of the certificates, and show that the certificate is not validated and the trusted channel is not established.

FCS\_DTLSC\_EXT.1.4

1. Test 1: The evaluator shall configure the server to perform an ECDHE key exchange in the DTLS connection using a non-supported curve (for example P-192) and shall verify that the TOE disconnects after receiving the server’s Key Exchange handshake message.

### FCS\_DTLSC\_EXT.2 Extended: DTLS Client Protocol with authentication

#### TSS

FCS\_DTLSC\_EXT.2.1

1. The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component.

FCS\_DTLSC\_EXT.2.2

1. The evaluator shall ensure that the TSS describes the client’s method of establishing all reference identifiers from the administrator/application-configured reference identifier, including which types of reference identifiers are supported (e.g Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the TOE.

FCS\_DTLSC\_EXT.2.4

1. The evaluator shall verify that TSS describes the Supported Elliptic Curves Extension and whether the required behaviour is performed by default or may be configured.

FCS\_DTLSC\_EXT.2.5

1. The evaluator shall ensure that the TSS description required per FIA\_X509\_EXT.2.1 includes the use of client-side certificates for DTLS mutual authentication.

FCS\_DTLSC\_EXT.2.6

1. The evaluator shall verify that the TSS describes the actions that take place if a message received from the DTLS Server fails the MAC integrity check.

FCS\_DTLSC\_EXT.2.7

1. The evaluator shall verify that TSS describes how replay is detected and silently discarded for DTLS records that have previously been received and too old to fit in the sliding window.

#### Guidance Documentation

FCS\_DTLSC\_EXT.2.1

1. The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that DTLS conforms to the description in the TSS.

FCS\_DTLSC\_EXT.2.2

1. The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in DTLS.

FCS\_DTLSC\_EXT.2.4

1. If the TSS indicates that the Supported Elliptic Curves Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves Extension.

FCS\_DTLSC\_EXT.2.5

1. The evaluator shall verify that the AGD guidance required per FIA\_X509\_EXT.2.1 includes instructions for configuring the client-side certificates for DTLS mutual authentication.

#### Tests

FCS\_DTLSC\_EXT.2.1

1. Test 1: The evaluator shall establish a DTLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level application protocol, e.g., as part of a syslog session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).
2. Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.
3. Test 3: The evaluator shall send a server certificate in the DTLS connection that the does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA ciphersuite.) The evaluator shall verify that the TOE disconnects after receiving the server’s Certificate handshake message.
4. Test 4: The evaluator shall configure the server to select the TLS\_NULL\_WITH\_NULL\_NULL ciphersuite and verify that the client denies the connection. Test 2 in FCS\_DTLSS\_EXT.1.1 or FCS\_DTLSS\_EXT.2.1 can be used as a substitute for this test.
5. Test 5: The evaluator perform the following modifications to the traffic:
6. Change the DTLS version selected by the server in the Server Hello to a non-supported DTLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.
7. Modify at least one byte in the server’s nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client’s Finished handshake message.
8. Modify the server’s selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.
9. Modify the signature block in the Server’s Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.
10. Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.
11. Send a garbled message from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.

FCS\_DTLSC\_EXT.2.2

1. The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a DTLS connection:
2. Test 1: The evaluator shall present a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. The evaluator shall verify that the connection fails.
3. Test 2: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.
4. Test 3: The evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contains the SAN extension. The evaluator shall verify that the connection succeeds.
5. Test 4: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.
6. Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier:
7. The evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.\*.example.com) and verify that the connection fails.
8. The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. \*.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.come) and verify that the connection fails.
9. Test 6: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.
10. Test 7: [conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

FCS\_DTLSC\_EXT.2.3

1. Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. If the certificate is validated and a trusted channel is established, the test passes. The evaluator then shall delete one of the certificates, and show that the certificate is not validated and the trusted channel is not established.

FCS\_DTLSC\_EXT.2.4

1. Test 1: The evaluator shall configure the server to perform an ECDHE key exchange in the DTLS connection using a non-supported curve (for example P-192) and shall verify that the TOE disconnects after receiving the server’s Key Exchange handshake message.

FCS\_DTLSC\_EXT.2.5

1. Test 1: The evaluator shall perform the following modification to the traffic:
2. Configure the server to require mutual authentication and then modify a byte in a CA field in the Server’s Certificate Request handshake message. The modified CA field must not be the CA used to sign the client’s certificate. The evaluator shall verify the connection fails.

FCS\_DTLSC\_EXT.2.6

1. Test 1: The evaluator shall establish a DTLS connection. The evaluator will then modify at least one byte in a record message, and verify that the Client discards the record or terminates the DTLS session.

FCS\_DTLSC\_EXT.2.7

1. Test 1: The evaluator shall set up a DTLS connection with a DTLS Server. The evaluator shall then capture traffic sent from the DTLS Server to the TOE. The evaluator shall retransmit copies of this traffic to the TOE in order to impersonate the DTLS Server. The evaluator shall observe that the TSF does not take action in response to receiving these packets and that the audit log indicates that the replayed traffic was discarded.

### FCS\_DTLSS\_EXT.1 Extended: DTLS Server Protocol

#### TSS

FCS\_DTLSS\_EXT.1.1

1. The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified are identical to those listed for this component.

FCS\_DTLSS\_EXT.1.3

1. The evaluator shall verify that the TSS describes how the DTLS Client IP address is validated prior to issuing a ServerHello message.

FCS\_DTLSS\_EXT.1.4

1. The evaluator shall verify that the TSS describes the key agreement parameters of the server key exchange message.

FCS\_DTLSS\_EXT.1.5

1. The evaluator shall verify that the TSS describes the actions that take place if a message received from the DTLS Client fails the MAC integrity check.

FCS\_DTLSS\_EXT.1.6

1. The evaluator shall verify that TSS describes how replay is detected and silently discarded for DTLS records that have previously been received and too old to fit in the sliding window.

#### Guidance Documentation

FCS\_DTLSS\_EXT.1.1

1. The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that DTLS conforms to the description in the TSS (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

FCS\_DTLSS\_EXT.1.4

1. The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

#### Tests

FCS\_DTLSS\_EXT.1.1

1. Test 1: The evaluator shall establish a DTLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level application protocol, e.g., as part of a syslog session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).
2. Test 2: The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server’s ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS\_NULL\_WITH\_NULL\_NULL ciphersuite and verify that the server denies the connection.
3. Test 3: The evaluator shall use a client to send a key exchange message in the TLS connection that the does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after the receiving the key exchange message.
4. Test 4: The evaluator shall perform the following modifications to the traffic:
5. Modify at a byte in the client’s nonce in the Client Hello handshake message, and verify that the server rejects the client’s Certificate Verify handshake message (if using mutual authentication) or that the server denies the client’s Finished handshake message.
6. Modify the signature block in the Client’s Key Exchange handshake message, and verify that the server rejects the client’s Certificate Verify handshake message (if using mutual authentication) or that the server denies the client’s Finished handshake message.
7. Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.
8. After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection.
9. Send a garbled message from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection.

FCS\_DTLSS\_EXT.1.3

1. Modify at least one byte in the cookie from the Server's HelloVerifyRequest message, and verify that the Server rejects the Client's handshake message.

FCS\_DTLSS\_EXT.1.4

1. The evaluator shall attempt a connection using an ECDHE ciphersuite and a configured curve and, using a packet analyzer, verify that the key agreement parameters in the Key Exchange message are the ones configured. (Determining that the size matches the expected size for the configured curve is sufficient.) The evaluator shall repeat this test for each supported NIST Elliptic Curve and each supported Diffie-Hellman key size.

FCS\_DTLSS\_EXT.1.5

1. The evaluator shall establish a connection using a client. The evaluator will then modify at least one byte in a record message, and verify that the Server discards the record or terminates the DTLS session.

FCS\_DTLSS\_EXT.1.6

1. The evaluator shall set up a DTLS connection. The evaluator shall then capture traffic sent from the DTLS Client to the TOE. The evaluator shall retransmit copies of this traffic to the TOE in order to impersonate the DTLS Client. The evaluator shall observe that the TSF does not take action in response to receiving these packets and that the audit log indicates that the replayed traffic was discarded.

### FCS\_DTLSS\_EXT.2 Extended: DTLS Server Protocol with mutual authentication

#### TSS

FCS\_DTLSS\_EXT.2.1

1. The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified are identical to those listed for this component.

FCS\_DTLSS\_EXT.2.3

1. The evaluator shall verify that the TSS describes how the DTLS Client IP address is validated prior to issuing a ServerHello message.

FCS\_DTLSS\_EXT.2.4

1. The evaluator shall verify that the TSS describes the key agreement parameters of the server key exchange message.

FCS\_DTLSS\_EXT.2.5

1. The evaluator shall verify that the TSS describes the actions that take place if a message received from the DTLS Client fails the MAC integrity check.

FCS\_DTLSS\_EXT.2.6

1. The evaluator shall verify that TSS describes how replay is detected and silently discarded for DTLS records that have previously been received and too old to fit in the sliding window.

FCS\_DTLSS\_EXT.2.7

1. The evaluator shall ensure that the TSS description required per FIA\_X509\_EXT.2.1 includes the use of client-side certificates for DTLS mutual authentication.

FCS\_DTLSS\_EXT.2.8

1. The evaluator shall verify that the TSS describes how the DN or SAN in the certificate is compared to the expected identifier.

#### Guidance Documentation

FCS\_DTLSS\_EXT.2.1

1. The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that DTLS conforms to the description in the TSS (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

FCS\_DTLSS\_EXT.2.3

1. The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

FCS\_DTLSS\_EXT.2.4

1. The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

FCS\_DTLSS\_EXT.2.7

1. The evaluator shall verify that the AGD guidance required per FIA\_X509\_EXT.2.1 includes instructions for configuring the client-side certificates for DTLS mutual authentication.

FCS\_DTLSS\_EXT.2.8

1. If the DN is not compared automatically to the Domain Name or IP address, username, or email address, then the evaluator shall ensure that the AGD guidance includes configuration of the expected DN or the directory server for the connection.

#### Tests

FCS\_DTLSS\_EXT.2.1

1. Test 1: The evaluator shall establish a DTLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level application protocol, e.g., as part of a syslog session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).
2. Test 2: The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server’s ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS\_NULL\_WITH\_NULL\_NULL ciphersuite and verify that the server denies the connection.
3. Test 3: The evaluator shall use a client to send a key exchange message in the DTLS connection that the does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after the receiving the key exchange message.
4. Test 4: The evaluator shall perform the following modifications to the traffic:
5. Modify at a byte in the client’s nonce in the Client Hello handshake message, and verify that the server rejects the client’s Certificate Verify handshake message (if using mutual authentication) or that the server denies the client’s Finished handshake message.
6. Modify the signature block in the Client’s Key Exchange handshake message, and verify that the server rejects the client’s Certificate Verify handshake message (if using mutual authentication) or that the server denies the client’s Finished handshake message.
7. Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.
8. After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection.
9. Send a garbled message from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection.

FCS\_DTLSS\_EXT.2.3

1. Modify at least one byte in the cookie from the Server's HelloVerifyRequest message, and verify that the Server rejects the Client's handshake message.

FCS\_DTLSS\_EXT.2.4

1. The evaluator shall attempt a connection using an ECDHE ciphersuite and a configured curve and, using a packet analyzer, verify that the key agreement parameters in the Key Exchange message are the ones configured. (Determining that the size matches the expected size for the configured curve is sufficient.) The evaluator shall repeat this test for each supported NIST Elliptic Curve and each supported Diffie-Hellman key size.

FCS\_DTLSS\_EXT.2.5

1. The evaluator shall establish a connection using a client. The evaluator will then modify at least one byte in a record message, and verify that the Server discards the record or terminates the DTLS session.

**FCS\_DTLSS\_EXT.2.6**

1. The evaluator shall set up a DTLS connection. The evaluator shall then capture traffic sent from the DTLS Client to the TOE. The evaluator shall retransmit copies of this traffic to the TOE in order to impersonate the DTLS Client. The evaluator shall observe that the TSF does not take action in response to receiving these packets and that the audit log indicates that the replayed traffic was discarded.

FCS\_DTLSS\_EXT.2.7 and FCS\_DTLSS\_EXT.2.8

1. Test 1: The evaluator shall configure the server to send a certificate request to the client and shall attempt a connection without sending a certificate from the client. The evaluator shall verify that the connection is denied.
2. Test 2: The evaluator shall configure the server to send a certificate request to the client without the supported\_signature\_algorithm used by the client’s certificate. The evaluator shall attempt a connection using the client certificate and verify that the connection is denied.
3. Test 3: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.
4. Test 4: The evaluator shall configure the client to send a certificate that does not chain to one of the Certificate Authorities (either a Root or Intermediate CA) in the server’s Certificate Request message. The evaluator shall verify that the attempted connection is denied.
5. Test 5: The evaluator shall configure the client to send a certificate with the Client Authentication purpose in the extendedKeyUsage field and verify that the server accepts the attempted connection. The evaluator shall repeat this test without the Client Authentication purpose and shall verify that the server denies the connection. Ideally, the two certificates should be identical except for the Client Authentication purpose.
6. Test 6: The evaluator shall perform the following modifications to the traffic:
7. Configure the server to require mutual authentication and then modify a byte in the client’s certificate. The evaluator shall verify that the server rejects the connection.
8. Configure the server to require mutual authentication and then modify a byte in the client’s Certificate Verify handshake message. The evaluator shall verify that the server rejects the connection.

FCS\_DTLSS\_EXT.2.9

1. The evaluator shall send a client certificate with an identifier that does not match an expected identifier and verify that the server denies the connection.

# Extended Component Definitions

## FCS\_DTLSC\_EXT DTLS Client Protocol

**Family Behaviour**

The component in this family addresses the ability for a client to use DTLS to protect data between the client and a server using the DTLS protocol.

**Component levelling**

2 FIA\_X509\_EXT.1 Certificate Authentication requires the TSF to check and validate certificates in accordance with RFC’s.

1 FIA\_X509\_EXT.1 Certificate Authentication requires the TSF to check and validate certificates in accordance with RFC’s.

FCS\_DTLSC\_EXT DTLS Client Protocol

FCS\_DTLSC\_EXT.1 DTLS Client requires that the client side of DTLS be implemented as specified.

FCS\_DTLSC\_EXT.2 DTLS Client requires that the client side of the DTLS implementation include mutual authentication.

**Management: FCS\_DTLSC\_EXT.1, FCS\_DTLSC\_EXT.2**

The following actions could be considered for the management functions in FMT:

1. There are no management activities foreseen.

**Audit: FCS\_DTLSC\_EXT.1, FCS\_DTLSC\_EXT.2**

The following actions should be auditable if FAU\_GEN Security audit data generation is included in the PP/ST:

1. Failure of DTLS session establishment.
2. DTLS session establishment
3. DTLS session termination

**FCS\_DTLSC\_EXT.1 DTLS Client Protocol**

Hierarchical to: No other components

Dependencies: FCS\_CKM. 1DataEncryption1 Cryptographic Key Generation

FCS\_CKM.2 Cryptographic Key Establishment

FCS\_COP.1/DataEncryption Cryptographic operation (AES Data encryption/decryption)

FCS\_COP.1/SigGen1SigGen Cryptographic operation (Signature Generation and Verification)

FCS\_COP.1/Hash Cryptographic operation (Hash Algorithm)

FCS\_COP.1/KeyedHash Cryptographic operation (Keyed Hash Algorithm)

FCS\_RBG\_EXT.1 Random Bit Generation

**FCS\_DTLSC\_EXT.1.1** The TSF shall implement [selection: DTLS 1.2 (RFC 6347), DTLS 1.0 (RFC 4347)] supporting the following ciphersuites:

* *[*selection: *Optional Ciphersuites:*
  + *[assignment: List of optional ciphersuites and reference to RFC in which each is defined]*
  + *no other ciphersuite]].*

Note that TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA is required in order to ensure compliance with RFC 5246.

**FCS\_DTLSC\_EXT.1.2** The TSF shall verify that the presented identifier matches the reference identifier according to RFC 6125 section 6.

The rules for verification of identify are described in Section 6 of RFC 6125. The reference identifier is established by the user (e.g. entering a URL into a web browser or clicking a link), by configuration (e.g. configuring the name of a mail server or authentication server), or by an application (e.g. a parameter of an API) depending on the application service. Based on a singular reference identifier’s source domain and application service type (e.g. HTTP, SIP, LDAP), the client establishes all reference identifiers which are acceptable, such as a Common Name for the Subject Name field of the certificate and a (case-insensitive) DNS name, URI name, and Service Name for the Subject Alternative Name field. The client then compares this list of all acceptable reference identifiers to the presented identifiers in the DTLS server’s certificate.

**FCS\_DTLSC\_EXT.1.3** The TSF shall only establish a trusted channel if the peer certificate is valid.

If DTLS is selected in FTP\_ITC then validity is determined by the identifier verification, certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity is tested in accordance with testing performed for FIA\_X509\_EXT.1/Rev. If DTLS is selected in FPT\_ITT then certificate validity is tested in accordance with testing performed for FIA\_X509\_EXT.1/ITT.

**FCS\_DTLSC\_EXT.1.4** The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [*assignment:* *List of supported curves including an option for ‘none’*].

If ciphersuites with elliptic curves were selected in FCS\_DTLSC\_EXT.1.1, a selection of one or more curves is required. If no ciphersuites with elliptic curves were selected in FCS\_DTLS\_EXT.1.1, then ‘none’ should be selected.

*This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS\_COP.1//SigGen and FCS\_CKM.1 and FCS\_CKM.2. This extension is required for clients supporting Elliptic Curve ciphersuites.*

**FCS\_DTLSC\_EXT.2 DTLS Client Protocol with Authentication**

Hierarchical to: FCS\_DTLSC\_EXT.1 DTLS Client Protocol

Dependencies: FCS\_CKM.1/DataEncryption Cryptographic Key Generation

FCS\_CKM.2 Cryptographic Key Establishment

FCS\_COP.1/DataEncryption Cryptographic operation (AES Data encryption/decryption)

FCS\_COP.1/SigGen Cryptographic operation (Signature Generation and Verification)

FCS\_COP.1/Hash Cryptographic ooperation (Hash Algorithm)

FCS\_COP.1/KeyedHash Cryptographic operation (Keyed Hash Algorithm)

FCS\_RBG\_EXT.1 Random Bit Generation

**FCS\_DTLSC\_EXT.2.1** The TSF shall implement [selection: DTLS 1.2 (RFC 6347), DTLS 1.0 (RFC 4347)] supporting the following ciphersuites:

* *[*selection: *Optional Ciphersuites:*
  + *[assignment: List of optional ciphersuites and reference to RFC in which each is defined]*
  + *no other ciphersuite]].*

The ST author should select the optional ciphersuites that are supported. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. The Suite B algorithms listed above (RFC 6460) are the preferred algorithms for implementation. TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA is required in order to ensure compliance with RFC 5246.

These requirements will be revisited as new DTLS versions are standardized by the IETF.

In a future version of this cPP DTLS v1.2 will be required for all TOEs.

**FCS\_DTLSC\_EXT.2.2** The TSF shall verify that the presented identifier matches the reference identifier according to RFC 6125.

The rules for verification of identify are described in Section 6 of RFC 6125. The reference identifier is established by the user (e.g. entering a URL into a web browser or clicking a link), by configuration (e.g. configuring the name of a mail server or authentication server), or by an application (e.g. a parameter of an API) depending on the application service. Based on a singular reference identifier’s source domain and application service type (e.g. HTTP, SIP, LDAP), the client establishes all reference identifiers which are acceptable, such as a Common Name for the Subject Name field of the certificate and a (case-insensitive) DNS name, URI name, and Service Name for the Subject Alternative Name field. The client then compares this list of all acceptable reference identifiers to the presented identifiers in the DTLS server’s certificate.

**FCS\_DTLSC\_EXT.2.3** The TSF shall only establish a trusted channel if the peer certificate is valid.

If DTLS is selected in FTP\_ITC then validity is determined by the identifier verification, certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity is tested in accordance with testing performed for FIA\_X509\_EXT.1/Rev. If DTLS is selected in FPT\_ITT then certificate validity is tested in accordance with testing performed for FIA\_X509\_EXT.1/ITT.

**FCS\_DTLSC\_EXT.2.4** The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [*assignment:* *List of supported curves including an option for ‘none’*].

If ciphersuites with elliptic curves were selected in FCS\_DTLSC\_EXT.2.1, a selection of one or more curves is required. If no ciphersuites with elliptic curves were selected in FCS\_DTLS\_EXT.2.1, then ‘none’ should be selected.

*This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS\_COP.1/SigGen and FCS\_CKM.1 and FCS\_CKM.2. This extension is required for clients supporting Elliptic Curve ciphersuites.*

**FCS\_DTLSC\_EXT.2.5** The TSF shall support mutual authentication using X.509v3 certificates.

The use of X.509v3 certificates for TLS is addressed in FIA\_X509\_EXT.2.1. This requirement adds that this use must include the client must be capable of presenting a certificate to a DTLS server forDTLS mutual authentication.

**FCS\_DTLSC\_EXT.2.6** The TSF shall [selection: *terminate the DTLS session, silently discard the record*] if a message received contains an invalid MAC.

*The Message Authentication Code (MAC) is keyed hash function specified in FCS\_COP.1(4). The MAC is negotiated during DTLS handshake phase and is used to protect integrity of messages received from the sender during DTLS data exchange. If MAC verification fails the session must be terminated or the record must be silently discarded.*

**FCS\_DTLSC\_EXT.2.7** The TSF shall detect and silently discard replayed messages for:

* DTLS records previously received.
* DTLS records too old to fit in the sliding window.

Replay Detection is described in section 4.1.2.6 of DTLS 1.2 (RFC 6347) and section 4.1.2.5 of DTLS 1.1 (RFC 4347). For each received record, the receiver verifies the record contains a sequence number is within the sliding receive window and does not duplicate the sequence number of any other record received during the session.

*"*Silently *Discard" means the TOE discards the packet without further processing.*

## FCS\_DTLSS\_EXT DTLS Server Protocol

**Family Behaviour**

The component in this family addresses the ability for a server to use DTLS to protect data between a client and the server using the DTLS protocol.

**Component leveling**

2 FIA\_X509\_EXT.1 Certificate Authentication requires the TSF to check and validate certificates in accordance with RFC’s.

1 FIA\_X509\_EXT.1 Certificate Authentication requires the TSF to check and validate certificates in accordance with RFC’s.

FCS\_DTLSS\_EXT DTLS Server Protocol

FCS\_DTLSS\_EXT.1 DTLS Server requires that the server side of TLS be implemented as specified.

FCS\_DTLSS\_EXT.2: DTLS Server requires the mutual authentication be included in the DTLS implementation.

**Management: FCS\_DTLSS\_EXT.1, FCS\_DTLSS\_EXT.2**

The following actions could be considered for the management functions in FMT:

1. There are no management activities foreseen.

**Audit: FCS\_DTLSS\_EXT.1, FCS\_DTLSS\_EXT.2**

The following actions should be auditable if FAU\_GEN Security audit data generation is included in the PP/ST:

1. Failure of DTLS session establishment.
2. DTLS session establishment
3. DTLS session termination

**FCS\_DTLSS\_EXT.1 DTLS Server Protocol**

Hierarchical to: No other components

Dependencies: FCS\_CKM.1 Cryptographic Key Generation

FCS\_CKM.2 Cryptographic Key Establishment

FCS\_COP.1//DataEncryption Cryptographic operation (AES Data encryption/decryption)

FCS\_COP.1//SigGen Cryptographic operation (Signature Generation and Verification)

FCS\_COP.1/Hash Cryptographic ooperation (Hash Algorithm)

FCS\_COP.1/KeyedHash Cryptographic operation (Keyed Hash Algorithm)

FCS\_RBG\_EXT.1 Random Bit Generation

**FCS\_TLSS\_EXT.1.1** The TSF shall implement [selection: *DTLS 1.2 (RFC 6347), DTLS 1.0 (RFC 4347)*] supporting the following ciphersuites:

* *[*selection: *Optional Ciphersuites:*
  + *[assignment: List of optional ciphersuites and reference to RFC in which each is defined]*
  + *no other ciphersuite]].*

The ciphersuites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional ciphersuites that are supported. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA is required in order to ensure compliance with RFC 5246.

**FCS\_DTLSS\_EXT.1.2** The TSF shall deny connections from clients requesting *[assignment: list of protocol versions]*.

In a future version of this cPP DTLS v1.0 will be required to be denied for all TOEs.

**FCS\_DTLSS\_EXT.1.3** The TSF shall not proceed with a connection handshake attempt if the DTLS Client fails validation.

*The process to validate the IP address of a DTLS client is specified in section 4.2.1 of RFC 6347 (DTLS 1.2) and RFC 4347 (DTLS 1.1). The TOE validates the DTLS client during Connection Establishment (Handshaking) and prior to the TSF sending a Server Hello message. After receiving a ClientHello, the DTLS Server sends a HelloVerifyRequest along with a cookie. The cookie is a signed message using the keyed hash function specified in FCS\_COP.1(4). The DTLS Client then sends another ClientHello with the cookie attached. If the DTLS server successfully verifies the signed cookie, the Client is not using a spoofed IP address.*

**FCS\_DTLSS\_EXT.1.4** The TSF shall generate key establishment parameters using RSA with key size [selection: *2048 bits, 3072 bits, 4096 bits, no other size*] and [selection: *over NIST curves* [selection: *secp256r1, secp384r1, secp521r1] and no other curves; Diffie-Hellman parameters of size 2048 bits and* [selection: *3072 bits, no other size*]*; no other*].

If the ST lists a DHE or ECDHE ciphersuite in FCS\_DTLSS\_EXT.1.1, the ST must include the Diffie-Hellman or NIST curves selection in the requirement. FMT\_SMF.1 requires the configuration of the key agreement parameters in order to establish the security strength of the DTLS connection.

**FCS\_DTLSS\_EXT.1.5** The TSF shall [selection: *terminate the DTLS session, silently discard the record*] if a message received contains an invalid MAC.

*The Message Authentication Code (MAC) is keyed hash function specified in FCS\_COP.1(4). The MAC is negotiated during DTLS handshake phase and is used to protect integrity of messages received from the sender during DTLS data exchange. If MAC verification fails the session must be terminated or the record must be silently discarded.*

**FCS\_DTLSS\_EXT.1.6** The TSF shall detect and silently discard replayed messages for:

* DTLS records previously received.
* DTLS records too old to fit in the sliding window.

Replay Detection is described in section 4.1.2.6 of DTLS 1.2 (RFC 6347) and section 4.1.2.5 of DTLS 1.1 (RFC 4347). For each received record, the receiver verifies the record contains a sequence number is within the sliding receive window and does not duplicate the sequence number of any other record received during the session.

"Silently Discard" means the TOE discards the packet without further processing.

**FCS\_DTLSS\_EXT.2 DTLS Server Protocol with mutual authentication**

Hierarchical to: FCS\_DTLSS\_EXT.1 DTLS Server Protocol

Dependencies: FCS\_CKM.1 Cryptographic Key Generation

FCS\_CKM.2 Cryptographic Key Establishment

FCS\_COP.1//DataEncryption Cryptographic operation (AES Data encryption/decryption)

FCS\_COP.1//SigGen Cryptographic operation (Signature Generation and Verification)

FCS\_COP.1/Hash Cryptographic ooperation (Hash Algorithm)

FCS\_COP.1/KeyedHash Cryptographic operation (Keyed Hash Algorithm)

FCS\_RBG\_EXT.1 Random Bit Generation

**FCS\_DTLSS\_EXT.2.1** The TSF shall implement [selection: *DTLS 1.2 (RFC 6347), DTLS 1.0 (RFC 4347)*] supporting the following ciphersuites:

* *[*selection: *Optional Ciphersuites:*
  + *[assignment: List of optional ciphersuites and reference to RFC in which each is defined]*
  + *no other ciphersuite]].*

The ciphersuites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional ciphersuites that are supported. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. The Suite B algorithms listed above (RFC 6460) are the preferred algorithms for implementation. TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA is required in order to ensure compliance with RFC 5246.

**FCS\_DTLSS\_EXT.2.2** The TSF shall deny connections from clients requesting *[assignment: list of protocol versions]*.

In a future version of this cPP DTLS v1.0 will be required to be denied for all TOEs.

**FCS\_DTLSS\_EXT.2.3** The TSF shall not proceed with a connection handshake attempt if the DTLS Client fails validation.

*The process to validate the IP address of a DTLS client is specified in section 4.2.1 of RFC 6347 (DTLS 1.2) and RFC 4347 (DTLS 1.1). The TOE validates the DTLS client during Connection Establishment (Handshaking) and prior to the TSF sending a Server Hello message. After receiving a ClientHello, the DTLS Server sends a HelloVerifyRequest along with a cookie. The cookie is a signed message using the keyed hash function specified in FCS\_COP.1(4). The DTLS Client then sends another ClientHello with the cookie attached. If the DTLS server successfully verifies the signed cookie, the Client is not using a spoofed IP address*.

**FCS\_DTLSS\_EXT.2.4** The TSF shall generate key establishment parameters using RSA with key size [selection: *2048 bits, 3072 bits, 4096 bits, no other size*] and [selection: *over NIST curves* [selection: *secp256r1, secp384r1] and no other curves; Diffie-Hellman parameters of size 2048 bits and* [selection: *3072 bits, no other size*]*; no other*].

If the ST lists a DHE or ECDHE ciphersuite in FCS\_TLSS\_EXT.2.1, the ST must include the Diffie-Hellman or NIST curves selection in the requirement. FMT\_SMF.1 requires the configuration of the key agreement parameters in order to establish the security strength of the DTLS connection.

**FCS\_DTLSS\_EXT.2.5** The TSF shall [selection: *terminate the DTLS connection, silently discard the record*] if a message received contains an invalid MAC.

*The Message Authentication Code (MAC) is negotiated during the DTLS handshake phase and is used to protect integrity of messages received from the sender during DTLS data exchange. If MAC verification fails the session must be terminated or silently discarded.*

**FCS\_DTLSS\_EXT.2.6**The TSF shall detect and silently discard replayed messages for:

* DTLS records that have previously been received.
* DTLS records too old to fit in the sliding window.

Replay Detection is described in section 4.1.2.6 of DTLS 1.2 (RFC 6347) and section 4.1.2.5 of DTLS 1.1 (RFC 4347). For each received record, the receiver verifies the record contains a sequence number is within the sliding receive window and does not duplicate the sequence number of any other record received during the session.

*"*Silently *Discard" means the TOE discards the packet without further processing.*

**FCS\_DTLSS\_EXT.2.7** The TSF shall support mutual authentication of DTLS clients using X.509v3 certificates.

**FCS\_DTLSS\_EXT.2.8** The TSF shall not establish a trusted channel if the peer certificate is invalid.

The use of X.509v3 certificates for DTLS is addressed in FIA\_X509\_EXT.2.1. This requirement adds that this use must include support for client-side certificates for DTLS mutual authentication.

If DTLS is selected in FTP\_ITC then validity is determined by the identifier verification, certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity is tested in accordance with testing performed for FIA\_X509\_EXT.1/Rev. If DTLS is selected in FPT\_ITT then certificate validity is tested in accordance with testing performed for FIA\_X509\_EXT.1/ITT.

**FCS\_DTLSS\_EXT.2.9** The TSF shall not establish a trusted channel if the distinguished name (DN) or Subject Alternative Name (SAN) contained in a certificate does not match the expected identifier for the peer.

The peer identifier may be in the Subject field or the Subject Alternative Name extension of the certificate. The expected identifier may either be configured, may be compared to the Domain Name, IP address, username, or email address used by the peer, or may be passed to a directory server for comparison. Matching should be performed by a bit-wise comparison.