Comment: Comment-0-
Comment: Comment-0-
Comment: Comment-0-

Functional Package for Transport Layer Security (TLS)

Version: 2.0-draft
2022-08-24
National Information Assurance Partnership

Revision History

Version	Date	Comment
1.0	2018-12-17	First publication
1.1	2019-03-01	Clarifications regarding override for invalid certificates, renegotiation_info extension, DTLS versions, and named Diffie-Hellman groups in DTLS contexts
2.0	2022-08-24	Added audit events, added TLS 1.3 support, deprecated TLS 1.0 and 1.1, updated algorithms/ciphersuites in accordance with CNSA suite RFC and to consider PSK, restructured SFRs for clarity

Contents

1 Introduction

1.1 Overview

Transport Layer Security (TLS) and the closely-related Datagram TLS (DTLS) are cryptographic protocols designed to provide communications security over IP networks. Several versions of the protocol are in widespread use in software that provides functionality such as web browsing, email, instant messaging, and voice-over-IP (VoIP). Major websites use TLS to protect communications to and from their servers. TLS is also used to protect communications between hosts and network infrastructure devices for administration. The underlying platform, such as an operating system, often provides the actual TLS implementation. The primary goal of the TLS protocol is to provide confidentiality and integrity of data transmitted between two communicating endpoints, as well as authentication of at least the server endpoint.

TLS supports many different methods for exchanging keys, encrypting data, and authenticating message integrity. These methods are dynamically negotiated between the client and server when the TLS connection is established. As a result, evaluating the implementation of both endpoints is typically necessary to provide assurance for the operating environment.

This "Functional Package for Transport Layer Security" (short name "TLS-PKG") defines functional requirements for the implementation of the TLS and DTLS protocols. The requirements are intended to improve the security of products by enabling their evaluation.

1.2 Terms

The following sections list Common Criteria and technology terms used in this document.

1.2.1 Common Criteria Terms

Assurance	Grounds for confidence that a TOE meets the SFRs [CC].
Base Protection Profile (Base-PP)	Protection Profile used as a basis to build a PP-Configuration.
Collaborative Protection Profile (cPP)	A Protection Profile developed by international technical communities and approved by multiple schemes.
Common Criteria (CC)	Common Criteria for Information Technology Security Evaluation (International Standard ISO/IEC 15408).
Common Criteria Testing Laboratory	Within the context of the Common Criteria Evaluation and Validation Scheme (CCEVS), an IT security evaluation facility accredited by the National Voluntary Laboratory Accreditation Program (NVLAP) and approved by the NIAP Validation Body to conduct Common Criteria-based evaluations.
Common Evaluation Methodology (CEM)	Common Evaluation Methodology for Information Technology Security Evaluation.
Distributed TOE	A TOE composed of multiple components operating as a logical whole.
Extended Package (EP)	A deprecated document form for collecting SFRs that implement a particular protocol, technology, or functionality. See Functional Packages.
Functional Package (FP)	A document that collects SFRs for a particular protocol, technology, or functionality.
Operational Environment (OE)	Hardware and software that are outside the TOE boundary that support the TOE functionality and security policy.
Protection Profile (PP)	An implementation-independent set of security requirements for a category of products.
Protection Profile Configuration (PP-Configuration)	A comprehensive set of security requirements for a product type that consists of at least one Base-PP and at least one PP-Module.

Protection Profile Module (PP-Module)	An implementation-independent statement of security needs for a TOE type complementary to one or more Base-PPs.
Security Assurance Requirement (SAR)	A requirement to assure the security of the TOE.
Security Functional Requirement (SFR)	A requirement for security enforcement by the TOE.
Security Target (ST)	A set of implementation-dependent security requirements for a specific product.
Target of Evaluation (TOE)	The product under evaluation.
TOE Security Functionality (TSF)	The security functionality of the product under evaluation.
TOE Summary Specification (TSS)	A description of how a TOE satisfies the SFRs in an ST.

1.2.2 Technical Terms

Certificate Authority (CA)	Issuer of digital certificates.
Datagram Transport Layer Security (DTLS)	Cryptographic network protocol, based on TLS, which provides communications security for datagram protocols.
Transport Layer Security (TLS)	Cryptographic network protocol for providing communications security over a TCP/IP network.

1.3 Compliant Targets of Evaluation

The Target of Evaluation (TOE) in this Package is a product which acts as a (D)TLS client, a (D)TLS server, or both. This Package describes the security functionality of TLS and DTLS in terms of [CC].

The contents of this Package must be appropriately combined with a PP or PP-Module. When this Package is instantiated by a PP or PP-Module, the Package must include selection-based requirements in accordance with the selections or assignments indicated in the PP or PP-Module. These may be expanded by the the ST author.

The PP or PP-Module which instantiates this Package must typically include the following components in order to satisfy dependencies of this Package. It is the responsibility of the PP or PP-Module author who instantiates this Package to ensure that dependence on these components is satisfied:

Component	Explanation
FCS_CKM.1	To support TLS ciphersuites that use RSA, DHE or ECDHE for key exchange, the PP or PP-Module must include FCS_CKM.1 and specify the corresponding key generation algorithm.
FCS_CKM.2	To support TLS ciphersuites that use RSA, DHE or ECDHE for key exchange, the PP or PP-Module must include FCS_CKM.2 and specify the corresponding algorithm.
FCS_COP.1	To support TLS ciphersuites that use AES for encryption and decryption, the PP or PP-Module must include FCS_COP.1 (iterating as needed) and specify AES with corresponding key sizes and modes. To support TLS ciphersuites that use SHA for hashing, the PP or PP-Module must include FCS_COP.1 (iterating as needed) and specify SHA with corresponding digest sizes.
FCS_RBG_EXT.1	To support random bit generation needed for the TLS handshake, the PP or PP-Module must include FCS_RBG_EXT.1.
FIA_X509_EXT.1	To support validation of certificates needed during TLS connection setup, the PP or PP-Module must include FIA_X509_EXT.1.
FIA_X509_EXT.2	To support the use of X509 certificates for authentication in TLS connection setup, the PP

or PP-Module must include FIA_X509_EXT.2.

An ST must identify the applicable version of the PP or PP-Module and this Package in its conformance claims.

2 Conformance Claims

Conformance Statement

An ST must claim exact conformance to this Package, as defined in the CC and CEM addenda for Exact Conformance, Selection-based SFRs, and Optional SFRs (dated May 2017).

CC Conformance Claims

This Package is conformant to Parts 2 (extended) and 3 (conformant) of Common Criteria Version 3.1, Revision 5.

PP Claim

This Package does not claim conformance to any Protection Profile.

Package Claim

This Package does not claim conformance to any packages.

Conformance Statement

This Package serves to provide Protection Profiles with additional SFRs and associated Evaluation Activities specific to TLS clients and servers.

This Package conforms to Common Criteria [CC] for Information Technology Security Evaluation, Version 3.1, Revision 5. It is CC Part 2 extended conformant.

In accordance with CC Part 1, dependencies are not included when they are addressed by other SFRs. The evaluation activities provide adequate proof that any dependencies are also satisfied.

3 Security Functional Requirements

This chapter describes the security requirements which have to be fulfilled by the product under evaluation. Those requirements comprise functional components from Part 2 of [CC]. The following conventions are used for the completion of operations:

- **Refinement** operation (denoted by **bold text** or ~~strikethrough text~~): Is used to add details to a requirement (including replacing an assignment with a more restrictive selection) or to remove part of the requirement that is made irrelevant through the completion of another operation, and thus further restricts a requirement.
- **Selection** (denoted by *italicized text*): Is used to select one or more options provided by the [CC] in stating a requirement.
- **Assignment** operation (denoted by *italicized text*): Is used to assign a specific value to an unspecified parameter, such as the length of a password. Showing the value in square brackets indicates assignment.
- **Iteration** operation: Is indicated by appending the SFR name with a slash and unique identifier suggesting the purpose of the operation, e.g. "/EXAMPLE1."

3.1 Auditable Events for Mandatory SFRs

The auditable events specified in this Functional Package are included in a Security Target if the incorporating PP or PP-Module supports audit event reporting through FAU_GEN.1 and all other criteria in the incorporating PP or PP-Module are met.

Table 1: Auditable Events for Mandatory Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FCS_TLS_EXT.1	No events specified	N/A

3.2 Cryptographic Support (FCS)

FCS_TLS_EXT.1 TLS Protocol

FCS_TLS_EXT.1.1

The TSF shall implement [**selection**:

- *TLS as a client*
- *TLS as a server*
- *DTLS as a client*
- *DTLS as a server*

].

Application Note: If *TLS as a client* is selected, then the ST must include the requirements from FCS_TLSC_EXT.1.
If *TLS as a server* is selected, then the ST must include the requirements from FCS_TLSS_EXT.1.

If *DTLS as a client* is selected, then the ST must include the requirements from FCS_DTLSC_EXT.1.
If *DTLS as a server* is selected, then the ST must include the requirements from FCS_DTLSS_EXT.1.

Evaluation Activities

FCS_TLS_EXT.1
TSS
The evaluator shall examine the TSS to verify that the TLS and DTLS claims are consistent with those selected in the SFR.

Guidance
The evaluator shall ensure that the selections indicated in the ST are consistent with selections in the dependent components.

Tests
There are no test activities for this SFR; the following information is provided as an overview of the expected functionality and test environment for all subsequent SFRs.

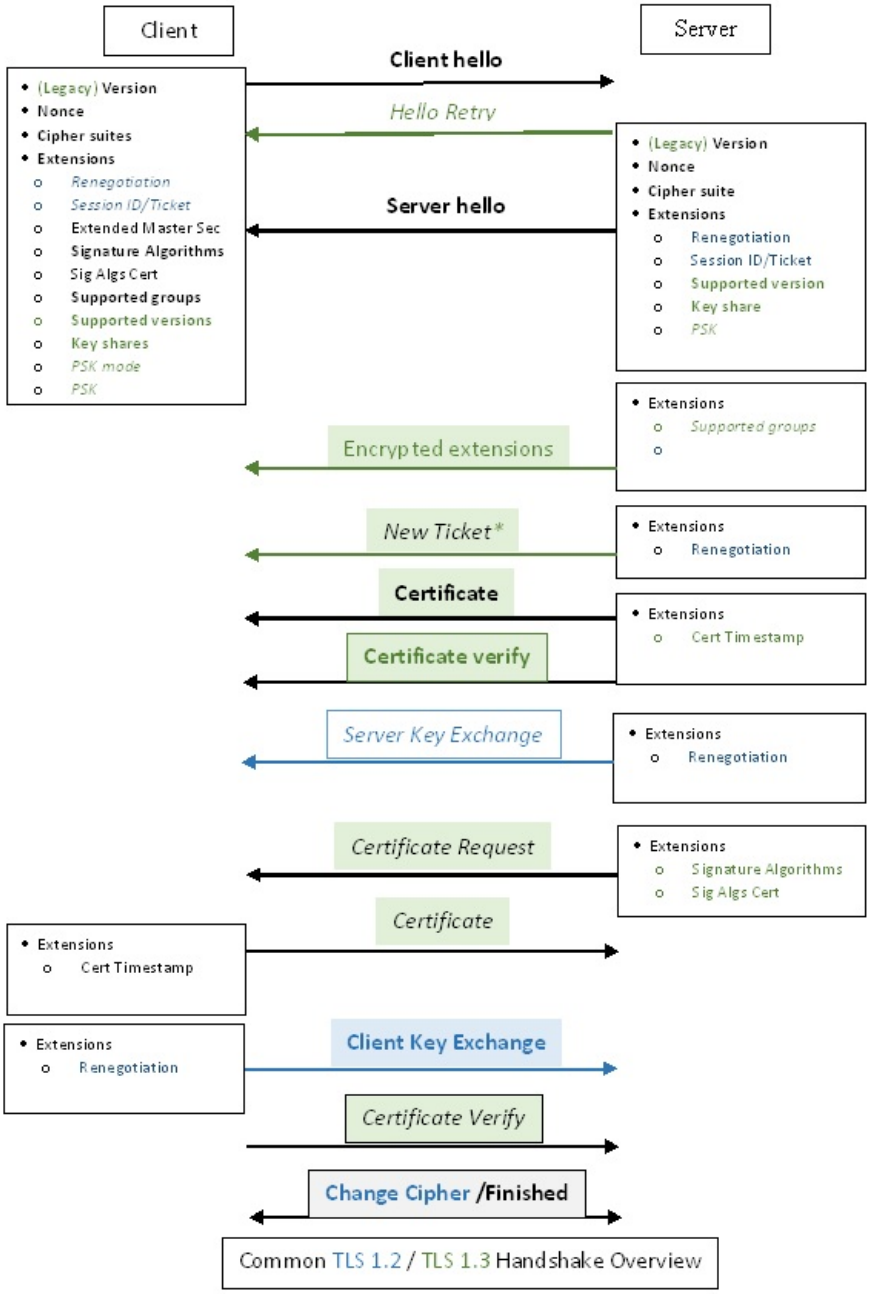

Figure 1: TLS Hello

The chart above provides an overview of the TLS hello messages, the content and protections, and the establishment of cryptographic keys in support of the protections.

- *Blue text indicates a message or content unique to TLS 1.2.*
- *Green text indicates uniqueness to TLS 1.3.*
- *Black text indicates features common to both TLS 1.2 and TLS 1.3.*
- *Bold text indicates mandatory features.*
- *Italics emphasizes optional features.*
- *A shaded text box indicates that the message is encrypted for TLS 1.2 (blue), TLS 1.3 (green) or both TLS 1.2 and TLS 1.3 (grey).*
- *An outlined text box indicates that the content in the message is signed, and/or provides authentication of the handshake to that point.*

Test Environment:

Tests for TLS 1.2 and TLS 1.3 include examination of the handshake messages and behavior of the TSF when presented with unexpected or invalid messages. For TLS 1.2 and below, previous versions of this Functional Package only required visibility of network traffic and the ability to modify a valid handshake message sent to the TSF.

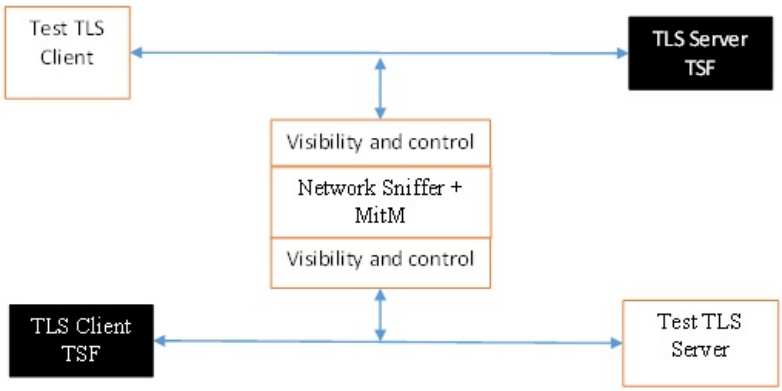

Figure 2: Test environment for TLS 1.2 using network traffic visibility and control tools

TLS 1.3 introduces the encryption of handshake messages subsequent to the server hello exchange which prevents visibility and control using midpoint capabilities. To achieve equivalent validation of TLS 1.3 requires the ability to modify the traffic underlying the encryption applied after the server hello message. This can be achieved by introducing additional control of the messages sent, and visibility of messages received by the test TLS client, when validating TLS server functionality or test server, when validating TLS client functionality.

Figure 3: Test environment for TLS 1.3 using custom endpoint capabilities for visibility and control

Typically, a compliant TLS 1.3 library modified to provide visibility and control of the handshake messages prior to encryption suffices for all tests. Such modification will require the test client and/or server to be validated.

Since validations of products supporting only TLS 1.2 are still expected under this Package, the test environment for TLS 1.2-only validations may include network sniffers and man-in-the-middle products that do not require such modifications to a compliant TLS 1.2 library. For consistency, a compliant TLS client (or TLS server) together with the network sniffers and man-in-the-middle capabilities will also be referred to as a test TLS client (or test TLS server, respectively) in the following evaluation activities.

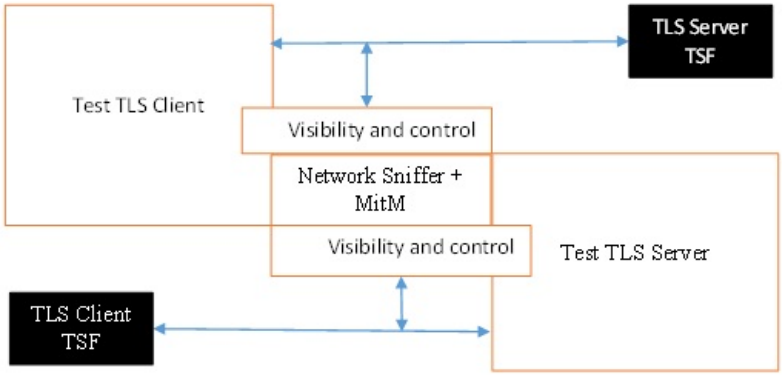

Figure 4: Combined test environment for TLS 1.2 and TLS 1.3 using both network tools and custom endpoint capabilities

FCS_TLSC_EXT.2 TLS Client Support for Mutual Authentication

This is a selection-based component. Its inclusion depends upon selection from .

FCS_TLSC_EXT.2.1

The TSF shall support mutual authentication using X.509v3 certificates during the handshake and [**selection**: *in support of post-handshake authentication requests, at no other time*], in accordance with [**selection**: *RFC 5246, section 7.4.4, RFC 8446, section 4.3.2*].

Application Note: Clients that support TLS 1.3 and post-handshake authentication claim ‘in support of post-handshake authentication requests’ in the first selection. The ‘at no other time’ selection is claimed for clients only supporting TLS 1.2 or for TLS 1.3 clients that do not support post-handshake authentication.

The certificate request sent by the server specifies the signature algorithms and certification authorities supported by the server. If the client does not possess a matching certificate, it sends an empty certificate message. The structure of the certificate request message is changed in TLS 1.3 to use the signature_algorithm, signature_algorithms_cert, and certificate_authorities extensions, and RFC 8446 allows for TLS 1.2 implementations to use the new message structure. The "RFC 8446, section 4.3.2" option is claimed in the second selection if TLS 1.3 is supported or if the RFC 8446 method is supported for TLS 1.2 servers. The "RFC 5246, section 7.4.4" option is claimed if the RFC 5246 method is supported for interoperability with TLS 1.2 servers that do not adopt the RFC 8446 method. When mutual authentication is supported, at least one of these methods must be claimed, per the selection.

This SFR is claimed if "mutual authentication" is selected in FCS_TLSC_EXT.1.1.

Evaluation Activities

FCS_TLSC_EXT.2
TSS
The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication. The evaluator shall also ensure that the TSS describes any factors beyond configuration that are necessary in order for the client to engage in mutual authentication using X.509v3 certificates.

Guidance
The evaluator shall ensure that the operational guidance includes any instructions necessary to configure the TOE to perform mutual authentication. The evaluator also shall verify that the operational guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

Tests
For each supported TLS version, the evaluator shall perform the following tests:

- ***Test 1.1:*** *The evaluator shall establish a TLS connection from the TSF to a test TLS server that negotiates the tested version and which is not configured for mutual authentication (i.e., does not send a Server’s Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE did not send a Client’s Certificate message (type 11) during handshake.*
- ***Test 1.2:*** *The evaluator shall establish a connection to a test TLS server with a shared trusted root that is configured for mutual authentication (i.e., it sends a Server’s Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE responds with a non-empty Client’s Certificate message (type 11) and Certificate Verify (type 15) message.*
- ***Test 1.3:*** *[conditional] If the TSF supports post-handshake authentication, the evaluator shall establish a pre-shared key between the TSF and a test TLS 1.3 server. The evaluator shall initiate a TLS session using the pre-shared key and confirm the TSF and test TLS 1.3 server successfully complete the TLS handshake and both support post-handshake authentication. After the session is successfully established, the evaluator shall initiate a certificate request message from the test TLS 1.3 server. The evaluator shall observe that the TSF receives that authentication request and shall take necessary actions, in accordance with the operational guidance, to complete the authentication request. The evaluator shall confirm that the test TLS 1.3 server receives certificate and certificate verification messages from the TSF over the channel that authenticates the client.*

Note: *TLS 1.3 certificate requests from the test server and client certificate and certificate verify messages are encrypted. The evaluator confirms that the TSF sends the appropriate messages by examining the messages received at the test TLS 1.3 server and by inspecting any relevant server logs. The evaluator may also take advantage of the calling application to demonstrate that the TOE receives data configured at the test TLS server.*

FCS_TLSC_EXT.3 TLS Client Downgrade Protection

This is a selection-based component. Its inclusion depends upon selection from .

FCS_TLSC_EXT.3.1

The TSF shall not establish a TLS channel if the server hello message includes [**selection**: *TLS 1.2 downgrade indicator, TLS 1.1 or below downgrade indicator*] in the server random field.

Application Note: The ST author claims the “TLS 1.2 downgrade indicator” when FCS_TLSC_EXT.1 indicates support for both TLS 1.3 and supplemental downgrade protection. This option is not claimed if TLS 1.3 is not supported. The “TLS 1.1 or below downgrade indicator” option may be claimed regardless of support for TLS 1.3, but should only be claimed if the TSF is capable of detecting the indicator. As indicated in FCS_TLSC_EXT.1.1, this FP requires the client to terminate TLS 1.1 or below sessions. It is acceptable for the TSF to always terminate TLS 1.1 sessions based on the server hello negotiated version field and ignore any downgrade indicator. However, a product that is capable of detecting the TLS 1.1 or below downgrade indicator may take different actions depending on whether the TLS 1.1 or below downgrade indicator is set.

This SFR is claimed if "supplemental downgrade protection" is selected in FCS_TLSS_EXT.1.1.

Evaluation Activities

FCS_TLSC_EXT.3
TSS
The evaluator shall review the TSS and confirm that the description of the TLS client protocol includes the downgrade protection mechanism in accordance with RFC 8446 and identifies any configurable features of the TSF needed to meet the requirements. If the ST claims that the TLS 1.1 and below indicator is processed, the evaluator shall confirm that the TSS indicates which configurations allow processing of the downgrade indicator and the specific response of the TSF when it receives the downgrade indicator as opposed to simply terminating the session for the unsupported version.

Guidance
The evaluator shall review the operational guidance and confirm that any instructions to configure the TSF to meet the requirements are included.

Tests
The evaluator shall perform the following tests to confirm the response to downgrade indicators from a test TLS 1.3 server:

- ***Test 2.1:*** *[conditional] If the TSF supports TLS 1.3, the evaluator shall initiate a TLS 1.3 session with a test TLS 1.3 server configured to send a compliant TLS 1.2 server hello (not including any TLS 1.3 extensions) but including the TLS 1.2 downgrade indicator ‘44 4F 57 4E 47 52 44 01’ in the last eight bytes of the server random field. The evaluator shall confirm that the TSF terminates the session.*

  Note: *It is preferred that the TSF send a fatal error alert message (e.g., illegal parameter), but it is acceptable that the TSF terminate the session without sending an error alert.*

- ***Test 2.2:*** *[conditional] If the TSF supports the TLS 1.1 or below downgrade indicator and if the ST indicates a configuration where the indicator is processed, the evaluator shall follow operational guidance instructions to configure the TSF so it parses a TLS 1.1 handshake to detect and process the TLS downgrade indicator. The evaluator shall initiate a TLS session between the TOE and a test TLS server that is configured to send a TLS 1.1 server hello message with the downgrade indicator ‘44 4F 57 4E 47 52 44 00’ in the last eight bytes of the server random field, but which is otherwise compliant with RFC 4346. The evaluator shall observe that the TSF terminates the session as described in the ST.*

  Note: *It is preferred that the TSF send a fatal error alert message (illegal parameter or unsupported version), but it is acceptable that the TSF terminate the session without sending an error alert.*

  Use of the TLS 1.1 and below indicator as a redundant mechanism where there is no

configuration that actually processes the value does not require additional testing, since this would be addressed by Test 2.1 for FCS_TLSC_EXT.1.1. This test is only required if the TSF responds differently (e.g., a different error alert) when the downgrade indicator is present than when TLS 1.1 or below is negotiated and the downgrade indicator is not present.

FCS_TLSC_EXT.4 TLS Client Support for Renegotiation

This is a selection-based component. Its inclusion depends upon selection from FCS_TLS_EXT.1.1.

FCS_TLSC_EXT.4.1

The TSF shall support secure renegotiation through use of [**selection**: *the "renegotiation_info" TLS extension, the TLS_EMPTY_RENEGOTIATION_INFO_SCSV signaling ciphersuite signaling value*] in accordance with RFC 5746, and shall terminate the session if an unexpected server hello is received and [**selection**: *hello request message is received, in no other case*].

Application Note: A client allowing TLS 1.2 connections may present either the "renegotiation_info" extension or the signaling ciphersuite value TLS_EMPTY_RENEGOTIATION_INFO_SCSV in the initial client hello message to indicate support for secure renegotiation. The ST author claims the methods supported. The TLS_EMPTY_RENEGOTIATION_INFO_SCSV is the preferred mechanism for TLS 1.2 protection against insecure renegotiation when the client does not renegotiate. The ST author will claim the 'hello request message is received' option in the second selection to indicate support for this mechanism.

RFC 5746 allows the client to accept connections with servers that do not support the extension; this FP refines RFC 5746 and requires the client to terminate sessions with such servers. Thus, unexpected server hello messages include an initial server hello negotiating TLS 1.2 that does not contain a renegotiation_info extension, an initial server hello negotiating TLS 1.2 that has a renegotiation_info that is non-empty, a subsequent server hello negotiating TLS 1.2 that does not contain a renegotiation_info extension, and a subsequent server hello negotiating TLS 1.2 that has a renegotiation_info extension with an incorrect renegotiated_connection value.

TLS 1.3 provides protection against insecure renegotiation by not allowing renegotiation. If TLS 1.3 is claimed in FCS_TLSC_EXT.1.1, the client receives a server hello that attempts to negotiate TLS 1.3, and the server hello also contains a renegotiation_info extension; the client will terminate the connection.

This SFR is claimed if "TLS as a client" is selected in FCS_TLS_EXT.1.1.

Evaluation Activities

FCS_TLSC_EXT.4
TSS
The evaluator shall examine the ST to ensure that TLS renegotiation protections are described in accordance with the requirements. The evaluator shall ensure that any configurable features of the renegotiation protections are identified.

Guidance
The evaluator shall examine the operational guidance to confirm that instructions for any configurable features of the renegotiation protection mechanisms are included.

Tests
The evaluator shall perform the following tests as indicated. One or both of Tests 1 or 2 is required, depending on whether the TSF is configurable to reject renegotiation or supports secure renegotiation methods defined for TLS 1.2. If TLS 1.3 is supported, Test 2 is required.

- ***Test 3.1:*** *[conditional] If the TSF supports a configuration to accept renegotiation requests for TLS 1.2, the evaluator shall follow any operational guidance to configure the TSF. The evaluator shall perform the following tests:*
  - ***Test 4.1.1:*** *The evaluator shall initiate a TLS connection with a test server configured to negotiate a compliant TLS 1.2 handshake. The evaluator shall inspect the messages received by the test TLS 1.2 server. The evaluator shall observe that either the "renegotiation_info" field or the SCSV ciphersuite is included in the ClientHello message during the initial handshake.*

- ***Test 4.1.2:*** *For each of the following sub-tests, the evaluator shall initiate a new TLS connection with a test TLS 1.2 server configured to send a renegotiation_info extension as specified, but otherwise complete a compliant TLS 1.2 session:*
  - ***Test 5.1.2.1:*** *The evaluator shall configure the test TLS 1.2 server to send a renegotiation_info extension whose value indicates a non-zero length. The evaluator shall confirm that the TSF terminates the connection.*

    Note: *It is preferred that the TSF sends a fatal error alert message (e.g., illegal parameter) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*
  - ***Test 5.1.2.2:*** *The evaluator shall configure the test TLS 1.2 server to send a compliant renegotiation_info extension and observe the TSF successfully completes the TLS 1.2 connection.*
  - ***Test 5.1.2.3:*** *The evaluator shall initiate a session renegotiation after completing a successful handhake with a test TLS 1.2 server that completes a successful TLS 1.2 handshake (as in Test 1.1) and then sends a hello reset request from the test TLS server with a "renegotiation_info" extension that has an unexpected "client_verify_data" or "server_verify_data" value (modify a byte from a compliant response). The evaluator shall verify that the TSF terminates the connection.*

    Note: *It is preferred that the TSF sends a fatal error alert message (e.g., illegal parameter, handshake error) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

- ***Test 5.2:*** *[conditional] if the TSF supports a configuration that prevents renegotiation, the evaluator shall perform the following tests:*
  - ***Test 6.2.1:*** *(TLS 1.2) [conditional] If the TLS supports a configuration to reject TLS 1.2 renegotiation, the evaluator shall follow the operational guidance as necessary to prevent renegotiation. The evaluator shall initiate a TLS session between the so-configured TSF and a test TLS 1.2 server that is configured to perform a compliant handshake, followed by a hello reset request. The evaluator shall confirm that the TSF completes the initial handshake successfully but terminates the TLS session after receiving the hello reset request.*

    Note: *It is preferred that the TSF sends a fatal error alert message (e.g., unexpected message) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*
  - ***Test 6.2.2:*** *[conditional] If the TSF supports TLS 1.3, the evaluator shall initiate a TLS session between the TSF and a test TLS 1.3 server that completes a compliant TLS 1.3 handshake, followed by a hello reset message. The evaluator shall observe that the TSF completes the initial TLS 1.3 handshake successfully, but terminates the session on receiving the hello reset message.*

    It is preferred that the TSF sends a fatal error alert message (e.g., unexpected message) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).

FCS_TLSC_EXT.5 TLS Client Support for Session Resumption

This is a selection-based component. Its inclusion depends upon selection from .

FCS_TLSC_EXT.5.1

The TSF shall support session resumption as a client via the use of [**selection**: *session ID in accordance with RFC 5246*, *tickets in accordance with RFC 5077*, *PSK and tickets in accordance with RFC 8446*].

Application Note: The ST author indicates which session resumption mechanisms are supported. One or both of the first two options, "session ID in accordance with RFC 5246" and "tickets in accordance with RFC 5077" are claimed for TLS 1.2 resumption. If resumption of TLS 1.3 sessions is supported, "PSK and tickets in accordance with RFC 8446" is selected, and the selection-based SFR FCS_TLSC_EXT.6 must also be claimed.

While it is possible to perform session resumption using PSK ciphersuites in TLS 1.2, this is uncommon. Validation of key exchange and session negotiation rules for PSK ciphersuites is independent of the source of the pre-shared key and is covered in FCS_TLSC_EXT.1.

This SFR is claimed if "session resumption" is selected in FCS_TLSC_EXT.1.1.

Evaluation Activities

FCS_TLSC_EXT.5
TSS
The evaluator shall examine the ST and confirm that the TLS client protocol description includes a description of the supported resumption mechanisms.

Guidance
The evaluator shall ensure the operational guidance describes instructions for any configurable features of the resumption mechanism.

Tests
The evaluator shall perform the following tests:

- ***Test 7.1:*** *For each supported TLS version and for each supported resumption mechanism that is supported for that version, the evaluator shall establish a new TLS session between the TSF and a compliant test TLS server that is configured to negotiate the indicated version and perform resumption using the indicated mechanism. The evaluator shall confirm that the TSF completes the initial TLS handshake and shall cause the TSF to close the session normally. The evaluator shall then cause the TSF to resume the session with the test TLS server using the indicated method and observe that the TSF successfully establishes the session.*

  Note: *For each method, successful establishment refers to proper use of the mechanism, to include compliant extensions and behavior, as indicated in the referenced RFC.*

- ***Test 7.2:*** *(TLS 1.3 session id echo) [conditional] If the TSF supports TLS 1.3, the evaluator shall initiate a new TLS 1.3 session with a test TLS server. The evaluator shall cause the test TLS server to send a TLS 1.3 server hello message (or a hello retry request if the TSF doesn't include the key share extension) that contains a different value in the legacy_session_id field, and observe that the TSF terminates the session.*

  Note: *It is preferred that the TSF sends a fatal error alert message (e.g., illegal parameter) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

FCS_TLSC_EXT.6 TLS Client TLS 1.3 Resumption Refinements

This is a selection-based component. Its inclusion depends upon selection from FCS_TLSC_EXT.5.1.

FCS_TLSC_EXT.6.1

The TSF shall send a psk_key_exchange_mode extension with the value psk_dhe when TLS 1.3 session resumption is offered.

FCS_TLSC_EXT.6.2

The TSF shall not send early data in TLS 1.3 sessions.

Application Note: This SFR is claimed when session resumption is supported for TLS 1.3. RFC 8446 allows pre-shared keys to be used directly and also allows early data to be protected using only the pre-shared key. This SFR refines the RFC to use PSK only with a supplemental DHE or ECDHE key exchange to ensure perfect forward secrecy for all sessions.

This SFR is claimed if "PSK and tickets in accordance with RFC 8446" is selected in FCS_TLSC_EXT.5.1.

Evaluation Activities

FCS_TLSC_EXT.6
TSS
The evaluator shall examine the TSS to verify that the TLS client protocol description indicates that the PSK exchange requires DHE mode and prohibits sending early data. The evaluator shall examine the TSS to verify it lists all applications that can be secured by TLS 1.3 using pre-shared keys and describes how each TLS 1.3 client application ensures data for the application is not sent using early data.

Guidance
The evaluator shall examine the operational guidance to verify that instructions for any configurable features that are required to meet the requirement are included. The evaluator shall ensure the operational guidance includes any instructions required to configure applications so the TLS 1.3 client implementation does not send early data.

Tests
[conditional] For each application that is able to be secured via TLS 1.3 using PSK, the evaluator shall follow operational guidance to configure the application not to send early data. The evaluator shall cause the application to initiate a resumed TLS 1.3 session between the TSF and a compliant test TLS 1.3 server as in Test 1 of FCS_TLSC_EXT.5. The evaluator shall observe that the TSF client hello for TLS 1.3 includes the psk_mode extension with the value PSK_DHE and sends a key share value for a supported group. The evaluator shall confirm that early data is not received by the test TLS server.

Note: *If no applications supported by the TOE provide data to TLS 1.3 that can be sent using PSK, this test is omitted.*

FCS_TLSS_EXT.1 TLS Server Protocol

This is a selection-based component. Its inclusion depends upon selection from FCS_TLS_EXT.1.1.

FCS_TLSS_EXT.1.1

The TSF shall implement TLS 1.2 (RFC 5246) and [**selection**: *TLS 1.3 (RFC 8446), no other TLS version*] as a server that supports additional functionality for session renegotiation protection and [**selection**:

- *mutual authentication*
- *supplemental downgrade protection*
- *session resumption*
- *no optional functionality*

] and shall reject connection attempts from clients supporting only TLS 1.1, TLS 1.0, or SSL versions.

Application Note: These requirements will be revisited as new TLS versions are standardized by the IETF.

If "mutual authentication" is selected, then the ST must additionally include the requirements from FCS_TLSS_EXT.2. If the TOE implements mutual authentication, this selection must be made.

If "session renegotiation protection" is selected, then the ST must additionally include the requirements from FCS_TLSS_EXT.4. If the TOE implements session renegotiation, or if TLS 1.3 is supported, this selection must be made.

If "supplemental downgrade protection" is selected, then the ST must additionally include the requirements from FCS_TLSS_EXT.3. If the TOE provides downgrade protection as indicated in RFC 8446, in particular, if TLS 1.3 is supported, this selection must be made.

If "session resumption" is selected, then the ST must additionally include the requirements from FCS_TLSS_EXT.5.

FCS_TLSS_EXT.1.2

The TSF shall be able to support the following TLS 1.2 ciphersuites: [**selection**:

- *TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289, RFC 8422*
- *TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289, RFC 8422*
- *TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288*
- *TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288*
- *TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289*
- *TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289*
- *TLS_RSA_WITH_AES_256_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_GCM_SHA256 as defined in RFC 5289*
- *TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289*
- *TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289*
- *TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289*
- *TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246*
- *TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246*
- *TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246*
- *PP-specific ciphersuites using pre-shared secrets including [**selection**:*
  - *TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384 as defined in RFC 8442*
  - *TLS_DHE_PSK_WITH_AES_256_GCM_SHA384 as defined in RFC 5487*
  - *TLS_RSA_PSK_WITH_AES_256_GCM_SHA384 as defined in RFC 5487*

- *TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256 as defined in RFC 8442*
- *TLS_DHE_PSK_WITH_AES_128_GCM_SHA256 as defined in RFC 5487*
- *TLS_RSA_PSK_WITH_AES_128_GCM_SHA256 as defined in RFC 5487*

]

- *the following TLS 1.3 ciphersuites: [**selection**:*
  - *TLS_AES_256_GCM_SHA384 as defined in RFC 8446*
  - *TLS_AES_128_GCM_SHA256 as defined in RFC 8446*
  - *[**assignment**: other TLS 1.3 ciphersuites]]*

]

] using a preference order based on [**selection**: *RFC 9151 priority, client hello ordering, [**assignment**: additional priority]*]

Application Note: The ST author should select the ciphersuites that are supported and must select at least one ciphersuite for each TLS version supported. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. If administrative steps need to be taken so that the ciphersuites negotiated by the implementation are limited to those in this requirement, then the appropriate instructions need to be contained in the guidance.

The final selection indicates the TOE's preference for negotiating a ciphersuite. RFC 9151 indicates the required ciphersuites for NSS systems and 'RFC 9151 priority' is claimed if those ciphersuites are selected whenever offered by the client. In general, it is preferred that GCM ciphersuites are selected over CBC ciphersuites, ECDHE is selected over RSA and DHE, and SHA256 or SHA384 is selected over SHA1.

The 'client hello ordering' option is claimed if client priority is considered; if both are claimed, the ST author should indicate which is primary and which is secondary, and whether the priority scheme is configurable. If other priority schemes or if tertiary priority is used, the ST author will claim the third option and describe the scheme in the ST.

Support for TLS_RSA_WITH_AES_128_CBC_SHA is not required despite being mandated by RFC 5246.

FCS_TLSS_EXT.1.3

The TSF shall not establish a connection with a client that does not indicate support for at least one of the supported ciphersuites.

FCS_TLSS_EXT.1.4

The TSF shall be able to process the following TLS client hello message extensions:

- signature_algorithms extension (RFC 8446) indicating support for [**selection**:
  - *ecdsa-secp384r1_sha384 (RFC 8446)*
  - *rsa_psks1_sha384 (RFC 8446)*

  ], and [**selection**:
  - *rsa_pss_pss_sha384 (RFC 8603)*
  - *rsa_pss_rsae_sha384 (RFC 8603)*
  - *[**assignment**: other non-deprecated signature algorithms]*
  - *no other signature algorithms*

  ]
- extended_master_secret extension (RFC 7627) enforcing client support
- the following other extensions: [**selection**:
  - *signature_algorithms_cert extension (RFC 8446) indicating support for [**selection**:*
    - *ecdsa-secp384r1_sha384 (RFC 8446)*
    - *rsk_psks1_sha384 (RFC 8446)*

    *], and [**selection**:*
    - *rsa_pss_pss_sha384 (RFC 8603)*
    - *rsa_pss_rsae_sha384 (RFC 8603)*
    - *rsa_pkcs1_sha256 (RFC 8446)*
    - *rsa_pss_rsae_sha256 (RFC 8446)*
    - *[**assignment**: other non-deprecated signature algorithms]*
    - *no other signature algorithms*

    ]
  - *supported_versions extension (RFC 8446) indicating support for TLS 1.3*

- *supported_groups extension (RFC 7919, RFC 8446) indicating support for [**selection**:*
  - *secp256r1*
  - *secp384r1*
  - *secp521r1*
  - *ffdhe2048(256)*
  - *ffdhe3072(257)*
  - *ffdhe4096(258)*
  - *ffdhe6144(259)*
  - *ffdhe8192(260)*

  ]
- *key_share extension (RFC 8446)*
- *no other extensions*

].

Application Note: If support for TLS 1.3 is claimed in FCS_TLSS_EXT.1.1, the selections for supported_versions, supported_groups, and key_share are claimed. Even if support for TLS 1.3 is not claimed, if ECDHE ciphersuites are claimed in FCS_TLSS_EXT.1.4, the entry for supported_groups is claimed. Support for additional extensions is acceptable. For signature_algorithms and signature_algorithms_certs (if supported), at least one of the signature schemes presented in the first sub-selection is claimed.

FCS_TLSS_EXT.1.5

The TSF shall perform key establishment for TLS using [**selection**:

- *RSA with size [**selection**: 2048 bits, 3072 bits, 4096 bits] and no other sizes*
- *Diffie-Hellman parameters with size [**selection**: 2048 bits, 3072 bits, 4096 bits, 6144 bits, 8192 bits] and no other sizes*
- *Diffie-Hellman groups [**selection**: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192] and no other groups, consistent with the client's supported groups extension and [**selection**: key share, no other] extension*
- *ECDHE parameters using elliptic curves [**selection**: secp256r1, secp384r1, secp521r1] and no other curves, consistent with the client's supported groups extension and [**selection**: key share, no other] extension and using non-compressed formatting for points*

].

Application Note: TLS 1.2 and TLS 1.3 perform key exchange using different mechanisms. In TLS 1.2, the requirements apply to the key exchange messages received by the server and optionally (for DHE or ECDHE ciphersuites) sent by the server. In TLS 1.3, the requirements apply to the values of the key share extension contained in the client and server hello messages. The options depend on the supported ciphersuites. For each session, the key exchange method is consistent with the selected ciphersuite (TLS 1.2), the supported groups extension (TLS 1.3 and conditionally, TLS 1.2), or the key share extension (TLS 1.3).

If the ST lists an RSA ciphersuite in FCS_TLSS_EXT.1.1, the ST must include the RSA selection in the requirement.

If the ST lists a DHE ciphersuite in FCS_TLSS_EXT.1.2, the ST must include the Diffie-Hellman selection for parameters of a certain size, the Diffie-Hellman groups selection in support of TLS 1.2 exchanges, or both. The selection for "Diffie-Hellman parameters" refers to the method defined by RFC 5246, section 7.4.3 where the server provides Diffie-Hellman parameters to the client. The “Diffie-Hellman groups” selection indicates key exchange negotiation in accordance with RFC 7919 using the supported groups extension. RFC 7919 identifies particular Diffie-Hellman groups, which are listed in the following selection. This option is the preferred mechanism for TLS 1.2, and must be claimed if TLS 1.3 DHE ciphersuites are supported.

If the ST lists an ECDHE ciphersuite in FCS_TLSS_EXT.1.2, the ST must include the selection for ECDHE using elliptic curves in the requirement, consistent with the support indicated for the supported groups extension in FCS_TLSS_EXT.1.4.

When TLS 1.3 is negotiated (if supported), the supported group negotiated (a supported DHE or ECDHE group) agrees with one of the client’s supported groups and the supplied key share element, and the product’s key share element is a member of the selected group. If the TLS 1.3 client does not initially provide a key share element for a group supported by both the product and the client, the TOE is expected to send a hello retry request message indicating the selected group; the requirement for matching the group indicated in the client’s

hello message applies to the client's hello message received in response to the hello retry request message.

This SFR is claimed if "TLS as a server" is selected in FCS_TLS_EXT.1.1.

Evaluation Activities

FCS_TLSS_EXT.1
TSS
The evaluator shall check the description of the implementation of this protocol in the TSS to ensure the supported TLS versions, features, ciphersuites, and extensions, are specified in accordance with RFC 5246 (TLS 1.2) and RFC 8446 (TLS 1.3 and updates to TLS 1.2) as appropriate. The evaluator shall check the description to see if beta TLS 1.3 versions are supported.

The evaluator shall verify that ciphersuites indicated in FCS_TLSS_EXT.1.2 are included in the description, and that none of the following ciphersuites are supported: ciphersuites indicating 'NULL,' 'RC2,' 'RC4,' 'DES,' 'IDEA,' or 'TDES' in the encryption algorithm component, indicating 'anon,' or indicating MD5 or SHA in the message digest algorithm component.

The evaluator shall verify that the TLS implementation description includes the extensions as required in FCS_TLSS_EXT.1.4.

The evaluator shall confirm that the TLS description includes the number and types of certificates that can be installed to represent the TOE.

Guidance
The evaluator shall check the operational guidance to ensure that it contains instructions on configuring the product so that the TSF conforms to the requirements. If the ST indicates that beta versions of TLS 1.3 are supported for backward compatibility, the evaluator shall ensure that the operational guidance provides instructions for disabling these versions.

The evaluator shall review the operational guidance to ensure instructions on installing certificates representing the TOE are provided.

Tests
The evaluator shall perform the following tests:

- ***Test 8.1:*** *(supported TLS 1.2 configurations) The evaluator shall perform the following tests:*
  - ***Test 9.1.1:*** *For each supported TLS 1.2 ciphersuite, the evaluator shall send a compliant TLS 1.2 client hello with the highest version or legacy version of 1.2 (value '03 03'), a single entry in the ciphersuites field consisting of the specific ciphersuite, and no supported version extension or key share extension. The evaluator shall observe the TSF's server hello indicates TLS 1.2 in the highest version or legacy version field, does not include a supported version or key share extension, and indicates the specific ciphersuite in the ciphersuite field. If the ciphersuite requires certificate-based authentication, the evaluator shall observe that the TSF sends a valid certificate representing the TOE and successfully completes the TLS handshake.*

    Note: *The ciphersuites TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384 as defined in RFC 8442, TLS_DHE_PSK_WITH_AES_256_GCM_SHA384 as defined in RFC 5487, TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256 as defined in RFC 8442, and TLS_DHE_PSK_WITH_AES_128_GCM_SHA256 as defined in RFC 5487, if supported, do not require certificate-based authentication of the server.*
  - ***Test 9.1.2:*** *(TLS 1.2 support for TLS 1.3 clients) [conditional] If the TSF is configurable to support only TLS 1.2 (or if TLS 1.3 is not supported), and if the TSF supports DHE or ECDHE ciphersuites, the evaluator shall follow any operational guidance instructions necessary to configure the TSF to only support TLS 1.2. For each supported TLS 1.2 ciphersuite with DHE or ECDHE indicated as the key exchange method, the evaluator shall send a client hello with the highest version or legacy version of 1.2 (value '03 03'), a list of ciphersuites consisting of one or more TLS 1.3 ciphersuites followed by the specific TLS 1.2 ciphersuite and no other TLS 1.2 ciphersuites in the ciphersuites field, and including a TLS 1.3 supported group and key share extension with consistent values. The evaluator shall observe that the TSF's server hello indicates TLS 1.2 in the highest version or legacy version field, does not include a supported version or key share extension, and indicates the specific TLS 1.2 ciphersuite in the ciphersuite field. The evaluator shall observe that the TSF completes the TLS 1.2 handshake successfully.*

    Note: *Supported ciphersuites using RSA key exchange should not be included in this test. The supported groups extension sent by the test TLS client should be consistent with the TLS 1.2 ciphersuite (e.g., it should be an EC group if the ciphersuite is ECDHE).*
  - ***Test 9.1.3:*** *(TLS 1.3 support) [conditional] If the TSF supports TLS 1.3, then for each*

supported TLS 1.3 ciphersuite and key exchange group, the evaluator shall send a compliant TLS 1.3 client hello indicating a list of one or more TLS 1.2 ciphersuites followed by the specific TLS 1.3 ciphersuite and no other ciphersuites in the ciphersuites field, a supported version extension indicating TLS 1.3 (value '03 04') only, a supported groups extension indicating the selected group, and a key share extension containing a value representing an element of the specific group. The evaluator shall observe the TSF's server hello contains the supported versions extension indicating TLS 1.3, the specific ciphersuite in the selected ciphersuite field, and a key share extension containing an element of the specific supported group. The evaluator shall observe that the TSF completes the TLS 1.3 handshake successfully.

***Note:** The connections in Test 1 may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP 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).

It is not necessary to pair every supported ciphersuite with every supported group. It is sufficient to use a set of ciphersuite and supported group pairs such that each ciphersuite and each supported group are included in this set.

TLS 1.3 includes the supported_groups extension in the encrypted_extensions message. This message may be observed at the test client after it is decrypted to help verify the key_share is actually a member of the supported group requested.

- ***Test 9.2:** (obsolete versions) The evaluator shall perform the following tests:*
  - ***Test 10.2.1:** For each of SSL version 2, SSL version 3, TLS version 1.0, and TLS version 1.1, the evaluator shall send a client hello to the TSF indicating the selected version as the highest version. The evaluator shall observe the TSF terminates the connection.*

    ***Note:** It is preferred that the TSF sends a fatal error alert message (e.g., protocol version, insufficient security) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*
  - ***Test 10.2.2:** The evaluator shall follow the operational guidance to configure the TSF to ensure any supported beta TLS 1.3 versions are disabled, as necessary. The evaluator shall send the TSF a client hello message indicating the supported version (referred to as the legacy version for TLS 1.3) with the value '03 04' and observe that the TSF responds with a server hello indicating the highest version supported.*

    ***Note:** Test 2.2 is intended to test the TSF response to non-standard versions, including beta versions of TLS 1.3. If the TSF supports such beta versions, the evaluator shall follow the operational guidance instructions to disable them prior to conducting Test 2.2.*

    Some TLS 1.3 implementations ignore the legacy version field and only check for the supported_versions extension to determine TLS 1.3 support by a client. It is preferred that the legacy version field should still be set to a standard version ('03 03') in the server hello, but it is acceptable that presence of the supported_versions indicating TLS 1.3 (value '03 04') overrides the legacy_version indication to determine highest supported version.
- ***Test 10.3:** (ciphersuites) The evaluator shall perform the following tests on handling unexpected ciphersuites using a test TLS client sending handshake messages compliant with the negotiated version except as indicated in the test:*
  - ***Test 11.3.1:** (ciphersuite not supported) For each supported version, the evaluator shall follow the operational guidance, if available, to configure the TSF to disable a supported ciphersuite. The evaluator shall send a compliant client hello to the TSF indicating support for the specific version and a ciphersuites field containing this single disabled ciphersuite. The evaluator shall observe that the TOE rejects the connection.*

    ***Note:** It is preferred that the TSF sends a fatal error alert message (e.g., handshake failure) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

    If the TSF's ciphersuites are not configurable, it is acceptable to use a named ciphersuite from the IANA TLS protocols associated with the tested version. Additional special cases of this test for special ciphersuites are performed separately.
  - ***Test 11.3.2:** (version confusion) For each supported version, the evaluator shall send a client hello that is compliant for the specific version that includes a list of ciphersuites consisting of a single ciphersuite not associated with that version. The evaluator shall observe that the TOE rejects the connection.*

    ***Note:** It is preferred that the TSF sends a fatal error alert message (e.g., handshake failure) in response to this, but it is acceptable that the TSF terminates the connection*

silently (i.e., without sending a fatal error alert).

It is preferred that Test 3.2 use TLS 1.3 ciphersuites for a server negotiating TLS 1.2. If TLS 1.3 is supported, Test 3.2 also includes a server negotiating TLS 1.3 with a TLS 1.2 ciphersuite – in this case, the negotiated ciphersuite should be chosen to be one supported by the TOE if negotiating TLS 1.2. If the TOE is configurable to allow both TLS 1.2 and TLS 1.3 clients (or does so by default), this configuration is used for both the TLS 1.2 and TLS 1.3 iteration of this test; otherwise the TOE is configured to support the negotiated version in each iteration.

- ***Test 11.3.3:*** *(null ciphersuite) For each supported version, the evaluator shall send a client hello indicating support for the version and include a ciphersuite list consisting of only the null ciphersuite (TLS_NULL_WITH_NULL_NULL, with the value '00 00') and observe that the TOE rejects the connection.*

  Note: *It is preferred that the TSF sends a fatal error alert message (e.g., handshake failure, insufficient security) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

- ***Test 11.3.4:*** *(anon ciphersuite) The evaluator shall send the TSF a TLS 1.2 handshake that is compliant, except that the ciphersuites field includes a ciphersuite list consisting only of ciphersuites using the anonymous server authentication method and observe that the TOE rejects the connection.*

  Note: *It is preferred that the TSF sends a fatal error alert message (e.g., handshake failure, insufficient security) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

  See IANA TLS parameters for available ciphersuites to be included in the client hello. The test ciphersuites list should include ciphersuites using supported cryptographic algorithms in as many of the other components as possible. For example, if the TSF supports the ciphersuite TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, the evaluator should include TLS_DH_ANON_WITH_AES_256_GCM_SHA_384.

- ***Test 11.3.5:*** *(deprecated encryption algorithm) The evaluator shall send the TSF a TLS 1.2 client hello that is compliant, except that the ciphersuites field is a list consisting only of ciphersuites indicating a deprecated encryption algorithm, including at least one each of NULL, RC2, RC4, DES, IDEA, and TDES. The evaluator shall observe that the TOE rejects the connection.*

  Note: *It is preferred that the TSF sends a fatal error alert message (e.g., handshake failure, insufficient security) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

  See IANA TLS parameters for available ciphersuites to be included. The test ciphersuite should use supported cryptographic algorithms for as many of the other components as possible. For example, if the TSF supports TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, the test could include TLS_ECDHE_PSK_WITH_NULL_SHA_384, TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 TLS_ECDHE_RSA_WITH_RC4_128_SHA, TLS_DHE_DSS_WITH_DES_CBC_SHA, TLS_RSA_WITH_IDEA_CBC_SHA, and TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA.

- ***Test 11.4:*** *(extensions)*
  - ***Test 12.4.1:*** *(signature algorithms) [conditional] If the TSF supports certificate-based authentication, then for each supported signature algorithm indicated in the ST, the evaluator shall perform the following sub-tests with certificates that represent the TOE. For each sub-test, the evaluator shall establish a to-be-signed certificate representing the TOE using a public-private key pair suitable for the specific signature algorithm value, and request that the certificate from a certification authority that uses the same signature algorithm, in accordance with FIA_X509_EXT.3.*

    the reference to FIA_X509_EXT.3 implies that any PP or module that uses this package must have the ability to generate its own CSRs (either as part of the TOE or its underlying platform), unsure if this is intended or if it will be permissible to load a cert issued by a CA.

    If the TSF also supports the signature_algorithms_cert extension, then for each value of the signature_algorithms_cert extension, the evaluator shall repeat the sub-tests using a to-be-signed certificate using a key pair consistent with the signature algorithm, with a certificate obtained from a certification authority that signs certificates using the specific value of the signature_algorithms_cert extension.

    Note: *The TSF supports certificate-based server authentication if the TLS 1.2 supported ciphersuites include ciphersuites other than TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384 as defined in RFC 8442, TLS_DHE_PSK_WITH_AES_256_GCM_SHA384 as defined in RFC 5487, TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256 as defined in RFC 8442, and TLS_DHE_PSK_WITH_AES_128_GCM_SHA256 as defined in RFC 5487. If these are the only supported ciphersuites, this test is omitted. For TLS 1.3, certificate-based server*

authentication, the client hello should not include the PSK extension.

The evaluator shall follow operational guidance instructions to provision the TSF with one or more of these certificates as indicated in the following sub-tests:

- ***Test 13.4.1.1:*** *(TLS 1.2) For each supported value of the signature_algorithm extension, the evaluator shall provision a certificate with a key pair compatible with the specific signature_algorithm value and send the TSF a TLS 1.2 client hello that indicates all supported ciphersuites and has a signature_algorithm extension consisting of a single value matching the specific signature algorithm.*

  If the TSF supports signature_algorithms_cert extension, the client hello also contains the value consistent with the previsioned certificate.

  The evaluator shall observe that the TSF negotiates TLS 1.2 with a TLS 1.2 ciphersuite that is compatible with the signature algorithm, and that it sends a certificate message containing the provisioned certificate with a key pair that is consistent with the specific signature_algorithm value (and signed using the signature_algorithms_cert extension value, if supported).

  Note: *For TLS 1.2, the ciphersuite describes the signature algorithm as RSA or ECDSA and is compatible with the certificate used if the signature algorithm component of the ciphersuite is of the same type as the signature value of the signature_algorithm extension.*

- ***Test 13.4.1.2:*** *[conditional] If the TSF supports TLS 1.3, then for each supported value of the signature_algorithm, the evaluator shall provision a certificate with a key pair that is compatible with the specific signature_algorithm value, send a TLS 1.3 client hello that indicates a supported ciphersuite and has a signature_algorithm extension consisting of a single value matching the specific signature algorithm.*

  If the TSF supports the signature_algorithms_cert extension, the client hello also contains a signature_algorithms_cert extension with a value consistent with the provisioned certificate.

  The evaluator shall observe that the TSF sends a certificate message containing the provisioned certificate consistent with the specific signature_algorithm value (and signed using the signature_algorithms_cert extension value) and a certificate verify message using the signature_algorithm extension value.

  Note: *For TLS 1.3, the certificate message and certificate verify is encrypted. The evaluator confirms the values of these messages as received at the test TLS client, using logs, or using a test TLS client designed to expose the certificates after they are decrypted.*

  It is not necessary to manually verify the signature used in the key exchange message (TLS 1.2) or certificate verify message (TLS 1.3).

- ***Test 13.4.1.3:*** *[conditional] If the ST indicates that the TSF supports provisioning of multiple certificates, the evaluator shall repeat Test 4.1.1 with both the provisioned certificate indicated for Test 4.1.1 (and 4.1.2 if TLS 1.3 is supported) and a certificate that does not match the signature_algorithm value. The evaluator shall observe that the certificate message (for TLS 1.2) does not include the certificate that does not match the signature_algorithm value (and signature_algorithms_cert value if supported) in the client hello.*

  [JF] Per SME, this test has issues with TLS 1.3 and will need updates

- ***Test 13.4.1.4:*** *(TLS 1.2) The evaluator shall provision a certificate as in Test 4.1.1 but shall send a client hello that only offers ciphersuites whose signature component does not match the value of the signature_algorithm extension. The evaluator shall observe that the TSF terminates the handshake.*

  Note: *It is preferred that the TSF sends a fatal error alert message (e.g., handshake failure, illegal parameter) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

- ***Test 13.4.2:*** *(extended master secret): The evaluator shall initiate a TLS 1.2 session with the TSF from a test TLS client for which the client hello does not include the extended master secret extension and observe that the TSF terminates the session.*

  Note: *It is preferred that the TSF sends a fatal error alert message (e.g., handshake error) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

- ***Test 13.5:*** *(key exchange) The evaluator shall perform the following tests to confirm compliant key exchange:*
  - ***Test 14.5.1:*** *(TLS 1.2 RSA key exchange) [conditional] If any of the supported TLS 1.2 ciphersuites in the ST includes RSA for the key exchange method, the evaluator shall perform the following sub-tests:*

- ***Test 15.5.1.1:*** *For each supported RSA key size, the evaluator shall provision the TSF with a valid certificate that has an RSA public key of that size. The evaluator shall initiate a valid TLS 1.2 handshake from a compliant test TLS 1.2 client and observe that the server certificate message matches the provisioned certificate.*
- ***Test 15.5.1.2:*** *For each supported RSA key size, the evaluator shall send the TSF a compliant TLS 1.2 client hello, but in place of the client's key exchange message, the evaluator shall send the TSF a (non-compliant) Key Exchange message that is properly formatted but uses an invalid EncryptedPreMasterSecret field in the TLS handshake (e.g., modify a byte of a properly computed value). The evaluator shall attempt to complete the handshake using compliant client change cipher spec and finished messages and verify that the TSF terminates the handshake in a manner that is indistinguishable from a finished message error and does not send application data.*

  Note: *Mitigations for oracle attacks described in RFC 5246 Appendix D require the TSF to exhibit the same behavior for key exchange failures as it does for finished message failures. It is preferred that the TSF send a fatal decrypt failure error alert at the end of the handshake in both this case and for a finished message error, but it is acceptable that the TSF terminate the session with another error alert, or without sending an error alert in either case. If the failure error alert is not for a decryption failure, the evaluator shall note that the TSF's response agrees with the response observed in the TLS 1.2 iteration of Test 5.2.*

- ***Test 15.5.2:*** *For each supported version, the evaluator shall initiate a compliant handshake up through the (implied for TLS 1.3) change cipher spec message. The evaluator shall then send a (non-compliant) client finished handshake message with an invalid 'verify data' value and verify that the server terminates the session and does not send any application data.*

  Note: *TLS 1.2 handshakes include explicit change cipher spec messages, but TLS 1.3 omits the change cipher spec message. If TLS 1.3 is supported, the modified finished message is sent as the final message from the client after receiving the server's second flight of handshake messages {encrypted extensions, (new ticket), (certificate, certificate verify), (certificate request)}.*

  It is preferred that the TSF send a fatal decryption failure error alert, but it is acceptable that the TSF terminate the session using another error alert or without sending an error alert.

  The finished message is encrypted. The invalid 'verify data' can be constructed by modifying a byte of a compliant finished message payload.

- ***Test 15.5.3:*** *(TLS 1.2 DHE or ECDHE key exchange) [conditional] If the ST indicates support for DHE or ECDHE ciphersuites for TLS 1.2, then the evaluator shall perform the following sub-tests:*
  - ***Test 16.5.3.1:*** *[conditional] If the TSF supports DHE ciphersuites and supports DHE parameters that are not specified in the supported groups extension, then for each supported DHE parameter set, the evaluator shall follow the operational guidance to configure the TSF to use the DHE parameters in its key exchange. The evaluator shall then initiate a TLS 1.2 handshake from a test client with a client hello indicating a single DHE ciphersuite. The evaluator shall observe that the TSF key exchange message indicates the configured parameters and ensure that the client key exchange is a valid point for the parameter set. The evaluator shall confirm that the TSF successfully completes the session.*

    The evaluator shall close the session and resend the client hello. After the TSF responds with a valid key exchange message, the evaluator shall send an empty client key exchange message and observe that the TSF terminates the session.

    Note: *It is preferred that the TSF sends a fatal error alert message (e.g., decryption failure, illegal parameter, handshake error) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

  - ***Test 16.5.3.2:*** *[conditional] If the TSF supports DHE ciphersuites and supports DHE groups in the supported groups extension, then for each supported DHE group, the evaluator shall send the TSF a compliant TLS 1.2 client hello indicating a single ciphersuite that is compatible with the group and indicating the group in the supported groups extension. The evaluator shall observe that the TSF negotiates TLS 1.2 using the indicated ciphersuite and that the server key exchange message indicates the specific group. The evaluator shall send the TOE a client key exchange with a valid point in the group and observe that the TSF successfully completes the session.*

    The evaluator shall close the session and resend the client hello. After the TSF responds with a valid key exchange message, the evaluator shall send the TSF a client key exchange with the public key value '0.' The evaluator shall observe that the TSF terminates the session.

The evaluator shall send a new client hello including the same ciphersuite but indicating a group not supported by the TSF in the supported groups extension. The evaluator shall observe that the TSF terminates the session.

Note: *It is preferred that the TSF sends a fatal error alert message (e.g., decryption failure, illegal parameter, handshake error) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

- ***Test 16.5.3.3:*** *[conditional] If the TSF supports ECDHE ciphersuites (and therefore supports ECDHE groups in the supported groups extension), the evaluator shall send a client hello message indicating a single supported ECDHE ciphersuite and including the supported ECDHE group in the supported groups extension. The evaluator shall observe that the TSF sends a key exchange message with a valid point of the specified group. The evaluator shall send the TSF a client key exchange message to the TSF consisting of a valid element in the supported group and observe that the TSF successfully completes the session.*

  The evaluator shall close the session and resend the client hello. After the TSF sends the valid key exchange message, the evaluator shall send a client key exchange message consisting of an invalid element of the supported group and observe that the TSF terminates the handshake.

  The evaluator shall send a third client hello to the TSF indicating the supported ECDHE ciphersuite and including an ECDHE group that is not supported. The evaluator shall observe that the TSF terminates the session.

  Note: *It is preferred that the TSF sends a fatal error alert message (e.g., decryption failure, illegal parameter, handshake error, insufficient security) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

  An invalid ECDSA point consists of properly formatted x and y components, but for which the equation of the curve is not satisfied. To obtain an invalid point, the evaluator can modify a byte of the y coordinate value of a valid point and confirm that the point is not on the curve.

  The IANA TLS parameters website lists registered ECDHE groups for use in selecting a non-supported group. If the TSF supports all registered ECDHE groups, it is acceptable to send the client hello without a supported groups extension. The TSF should reject such a client hello, but it is acceptable for the TSF to default to a supported group. In this case, the TSF passes the test.

- ***Test 16.5.4:*** *(TLS 1.3 key exchange) [conditional] If the TSF supports TLS 1.3, then for each supported group the evaluator shall perform the following sub-tests:*
  - ***Test 17.5.4.1:*** *The evaluator shall send the TSF a compliant TLS 1.3 client hello indicating a single key share value from the supported group and shall observe that the server hello includes valid elements of the supported group.*
  - ***Test 17.5.4.2:*** *The evaluator shall send the TSF a TLS 1.3 client hello indicating a supported groups value supported by the TSF but containing a key share extension indicating an element claiming to be in the supported group that does not represent a valid element of the group. The evaluator shall observe that the TSF terminates the session.*

    Note: *It is preferred that the TSF sends a fatal error alert message (e.g., illegal parameter, handshake failure, decryption failure) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

    For DHE groups, the invalid element may be of the wrong length; for ECDHE groups, the invalid element has coordinates (x and y) that do not satisfy the equation of the elliptic curve. To obtain an invalid ECDHE point, the evaluator can modify a byte of the y coordinate value of a valid point and confirm that the point is not on the curve.

- ***Test 17.5.5:*** *For each supported version, the evaluator shall initiate a TLS handshake from a test TLS client with compliant handshake messages negotiating the version and supported parameters to include the change cipher spec message (implied for TLS 1.3), but which omits the finished message and instead sends an application message containing random data. The evaluator shall observe that the TSF terminates the connection.*

  Note: *It is preferred that the TSF sends a fatal error alert message (e.g., decryption failure) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

  Application data is indicated by the TLSCipherText ContentType field having value 23 (application data). The legacy record version '03 03' and length fields should match a valid TLSCipherText message of the same size.

FCS_TLSS_EXT.2 TLS Server Support for Mutual Authentication

This is a selection-based component. Its inclusion depends upon selection from FCS_TLSS_EXT.1.1.

FCS_TLSS_EXT.2.1

The TSF shall support authentication of TLS clients using X.509v3 certificates during the TLS handshake and [**selection**: *during post-handshake requests*, *at no other time*] using the certificate types indicated in the client's signature_algorithms and [**selection**: *signature_algorithms_cert*, *no other*] extension.

Application Note: TLS 1.3 supports authentication after completing the abbreviated handshake with pre-shared keys. A server may send a client a certificate request after the finished message whenever the client includes the post-handshake authentication extension. The ST author claims 'during post-handshake request' if this feature is supported. If TLS 1.3 is not supported, or if the TLS post-handshake request extension is not recognized in a TLS 1.3 handshake, the ST author selects 'at no other time'.

FCS_TLSS_EXT.2.2

The TSF shall support authentication of TLS clients using X.509v3 certificates in accordance with FIA_X509_EXT.1.

FCS_TLSS_EXT.2.3

The TSF shall be able to reject the establishment of a trusted channel if the requested client certificate is invalid and [**selection**:

- *continue establishment of a server-only authenticated TLS channel in accordance with FCS_TLSS_EXT.1 in support of [**selection**: all applications, [**assignment**: list of calling applications that accept both authenticated and unauthenticated client sessions]] when an empty certificate message is provided by the client*
- *continue establishment of a mutually authenticated TLS channel when revocation status information for the [**selection**: client's leaf certificate, [**assignment**: specific intermediate leaf CA certificates], any non-trust store certificate in the certificate chain] is not available in support of [**selection**: all applications, [**assignment**: list of calling applications configurable to perform certificate status information bypass processing]] as [**selection**: configured by an administrator, confirmed by the application user, as a default for [**assignment**: subset of applications]]*
- *no other processing options for missing or invalid client certificates*

].

Application Note: The ST author claims any certificate processing exceptions that are allowed for specific calling applications. The 'continue establishment of a server-only authenticated TLS channel…' selection is claimed if the TLS product supports applications that can provide services to unauthenticated users if the user does not possess an appropriate certificate. Within this selection, the ST author indicates which applications are able to support both authenticated and unauthenticated users.

The ST author claims 'continue establishment of a mutually authenticated TLS channel…' if there is an administrator configuration or user confirmation that revocation status information is not available for one or more of the certificates in the client's certificate chain. If claimed, the ST author will describe in the assignment for intermediate values which CA certificates are included in the exception (for example, "all intermediates but the issuing CA" or "specific end-entity certificates as configured"). Within this selection, the ST author specifies which applications are impacted and which authorized user is allowed to approve continuing with the session when revocation information is not available. If an administrator configures whether a user may accept a certificate without status information, both selections are claimed. The 'as a default' should only be selected for applications that do not have access to revocation information. Methods for obtaining revocation information are included in FIA_X509_EXT.1.

FCS_TLSS_EXT.2.4

The TSF shall be able to [**selection**:

- *not establish a TLS session if an entry of the Distinguished Name or a [**selection**: rfc822_name, dns_name, [**assignment**: supported name types]] in the Subject Alternate Name extension contained in the client certificate*

*does not match one of the expected identifiers for the client in accordance with [**selection**: RFC 2822, RFC 6125, RFC 5280, [**assignment**: RFC for the supported name type]] matching rules*

- *pass the [**selection**: validated certificate, RFC 822 name normalized according to RFC 822, DNS name normalized according to RFC 6125, [**assignment**: list of RFC 5280 name types and normalization rules], [**assignment**: list of 'other' name types and standard normalization rules]] to [**assignment**: list of calling applications capable of making access decisions]*

].

Application Note: Authorization for services provided by the applications that are protected by the TLS session is determined either by the application establishing a set of reference identifiers or by passing the received identifiers to the application. The ST author indicates the methods supported and, for each method supported, indicates all name types supported; at least one name type is required. In the assignment of the first option, the ST author indicates all name types and the corresponding method for matching in the sub-selections. In the second method option, the ST author indicates which name type normalizations the product supports. If the product passes the entire validated certificate to the application, no normalization of the names contained in the certificate is expected.

If name normalization is claimed, care should be taken regarding wildcards and IP addresses. IP addresses embedded in DNS host names and in Directory Name CN components have been observed to include non-standard wildcard designations including the '*' character. Any embedded IP addresses should use standard CIDR notation and should not include nonstandard encoding.

This SFR is claimed if "mutual authentication" is selected in FCS_TLSS_EXT.1.1.

Evaluation Activities

FCS_TLSS_EXT.2
TSS
The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication, and that the description includes any certificate validation exception rules and the name types supported for matching to reference identifiers for all applications that use TLS. The evaluator shall examine the TSS to ensure that any CN-embedded name types that are used include a description of the encoding and matching rules.

Guidance
The evaluator shall verify that the operational guidance includes instructions for configuring trust stores for client-side certificates used in TLS mutual authentication. The evaluator shall ensure that the operational guidance includes instructions for configuring the server to require mutual authentication of clients using these certificates and for configuring any certificate validation exception rules. The evaluator shall ensure that the operational guidance includes instructions for configuring reference identifiers normalized or matched by the TSF and matching rules for the supported name types.

Tests
The evaluator shall use TLS as a function to verify that the validation rules in FIA_X509_EXT.1 are adhered to and shall perform the tests listed below. The evaluator shall apply the operational guidance to configure the server to require TLS mutual authentication of clients for these tests unless overridden by instructions in the test activity.

***Note:** TLS 1.3 is a fundamentally different protocol than TLS 1.2, so even though the certificate validation and name checking tests are identical for both versions, it is likely that early deployments of TLS 1.3 may use a different code-base that warrants independent testing. If TLS 1.3 is supported and the evaluator can verify that the TSF uses the same code-base for certificate validation and name checking for both TLS 1.3 and TLS 1.2, it is acceptable that testing be performed for only one version for these tests.*

- ***Test 18.1:** For each supported version, the evaluator shall follow the operational guidance to configure the TOE to require valid client authentication with no exceptions and initiate a TLS session from a compliant TLS test client supporting that version. The evaluator shall ensure that the test client sends a certificate_list structure which has a length of zero. The evaluator shall verify the TSF terminates the session and no application data flows.*

  ***Note:** It is preferred that the TSF sends a fatal error alert message (e.g., handshake failure, bad certificaate, unknown certificate, unknown CA) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

- ***Test 18.2:** [conditional] If the ST indicates that the TSF supports establishment of a TLS*

session for missing or invalid certificates, then for each supported version, and for each supported response option for a missing or invalid certificate indicated in FCS_TLSS_EXT.2.3, the evaluator shall configure the TSF according to the operational guidance to respond as indicated for the calling application. The evaluator shall send client handshake messages from a test TLS client as indicated for each sub-test. The evaluator shall perform the following sub-tests:

- ***Test 19.2.1:*** *[conditional]: If the TSF supports non-authenticated session establishment when receiving an empty certificate message, the evaluator shall initiate a TLS handshake from a compliant test TLS client supporting the version and providing a certificate message containing a certificate_list structure of length zero. The evaluator shall confirm that the TSF notifies the calling application that the user is unauthenticated.*

  Note: *Specific procedures for determining that the calling application is notified will vary based on the application. If an API to the calling application is not available, the evaluator may attempt to configure the calling application to provide a different response (e.g., require authentication for flagged data) for authenticated and non-authenticated users and make a request at the test client that results in a response indicating the application is treating the client as non-authenticated.*

- ***Test 19.2.2:*** *[conditional] If the TSF supports exceptions for when revocation status information is unavailable, then the evaluator shall follow the operational guidance to attempt to establish a narrowly defined exception for which both exempt and non-exempt certificates can be established. The evaluator shall establish a primary certificate chain for the test client that only exhibits the allowed exception and one or more alternate certificate chains for the test client that do not pass the exception rule, as necessary to test the boundaries of the exception rules.*

  The evaluator shall follow the operational guidance to remove any cached revocation status information for the test client's primary certificate chain. The evaluator shall initiate a valid TLS session from the test client that presents the primary certificate for the test client, provide any feedback requested by the TSF to confirm the exception, and observe that the TSF allows the certificate and completes the TLS handshake successfully.

  For each alternate certificate chain, the evaluator shall repeat the session initiation from the test client but present the alternate certificate chain and observe that the TSF terminates the session.

  Note: *It is preferred that the TSF sends a fatal error alert message (e.g., bad certificate, unknown certificate, access denied, handshake error) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

  The alternate certificate chains are intended to test the boundaries of the exception rules. For example, if the exception rule indicates that only leaf certificates are exempt, the evaluator will include an alternate certificate chain for which a CA certificate's revocation information is advertised but is not available; if the exception can be configured for an explicit leaf certificate, or particular subjects, an alternate chain will be included that does not include an excepted certificate or subject. If the exception rules can be configured for all certificates having advertised revocation information, an alternate certificate chain can include an expired certificate – only one additional validity failure (e.g., expired certificate) is required in this case. More comprehensive validity failure handling is addressed by testing for FIA_X509_EXT.1.

- ***Test 19.3:*** *For each supported version, the evaluator shall configure the TSF to negotiate the version and require client authentication and perform the following steps:*
  - *For each supported name matching method indicated in the outer selection of FCS_TLSS_EXT.2.4, and for each name type supported by the matching method as indicated in the inner-selections claimed in each outer selection, the evaluator shall establish a valid primary certificate chain with single names for a test client containing only the supported name types and a valid alternate certificate chain with single names indicating a different name of the same type.*
  - *[conditional] If any of the supported name types include CN encoding of a name type also supported as a SAN entry, the evaluator shall establish additional certificate chains as follows: [JF] This was updated based on SME feedback but unsure if further updates are needed*
    - *The evaluator shall establish a primary certificate chain with multiple names, to include a leaf certificate with:*
      - *a SAN entry that matches the name in the primary certificate chain with single names, of the same SAN name type; and*
      - *a CN entry encoding the same SAN type which matches the name in the alternate certificate chain with single names of the CN encoding of the same SAN name type.*
    - *The evaluator shall establish an alternate certificate chain with multiple names, to include a leaf certificate with:*

- *a SAN entry that matches the name in the alternate certificate chain with single names, of the same SAN name type; and*
- *a CN entry encoding the same SAN type which matches the name in the primary certificate chain with single names, of the CN encoding of the same SAN name type.*

In this case, the evaluator shall also obtain an alternate certificate chain with multiple names including a CN encoding of the name matching that in the corresponding primary certificate containing only the CN encoding and a SAN entry of the same type that matches the name in the alternate certificate chain having the same SAN type.

- *[conditional] If any of the supported name types include CN encoding, the evaluator shall follow the operational guidance to configure the TSF, establishing trust in the root CA for all primary and alternate certificate chains. The evaluator shall configure the TSF and any relevant TOE applications that use TLS for client authentication as necessary to establish reference identifiers that match the names in the client's primary certificate chains with single names, but not matching any of the names in the alternate certificate chains with single names.*
- *For each primary certificate chain (with single or multiple names), the evaluator shall initiate a TLS session from the test TLS client that is configured to present the primary certificate chain in a certificate message and a valid certificate verify message in response to the server's certificate request message. The evaluator shall confirm that the TSF accepts the certificate and completes the authenticated TLS session successfully.*
- *For each alternate certificate chain (with single or multiple names), the evaluator shall initiate a TLS session from the test TLS client that is configured to present the alternate certificate chain in a certificate message and a valid certificate verify message in response to the server's certificate request message. The evaluator shall confirm that the TSF terminates the session.*

***Note:** It is preferred that the TSF sends a fatal error alert message (e.g., access denied) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

The intent of this test is to confirm that for each method that the TSF uses to match name types presented in validated certificates, it is able to recognize both matching and non-matching names. Names of special types implicitly encoded in the CN entry of the certificate subject name are especially prone to error since they may only be validated by the issuing CA as a directory name (RDN) type, especially if the issuing CA is unaware of the intended encoding as a different name type. It is a best practice that when the CN is interpreted as an embedded name type other than RDN, an explicitly encoded SAN entry should take precedence.

FCS_TLSS_EXT.3 TLS Server Downgrade Protection

This is a selection-based component. Its inclusion depends upon selection from FCS_TLSS_EXT.1.1.

FCS_TLSS_EXT.3.1

The TSF shall set the server hello extension to a random value concatenated with the TLS 1.2 downgrade indicator when negotiating TLS 1.2 as indicated in RFC 8446 section 4.1.3.

Application Note: This SFR is claimed if the TSF supports TLS 1.3. RFC 8446 requires both the TLS 1.2 downgrade indicator as well as an indicator for TLS 1.1 and below. This FP requires the server to reject attempts to establish TLS 1.1 and below, making this mechanism redundant. However, products may still implement both indicators to be compliant with the RFC.

This SFR is claimed if "supplemental downgrade protection" is selected in FCS_TLSS_EXT.1.1.

Evaluation Activities

FCS_TLSS_EXT.3
TSS
The evaluator shall examine the ST and confirm that the TLS description includes details on the session downgrade protections that are supported.

Guidance
The evaluator shall examine the operational guidance to confirm that instructions are included to configure the TSF to support only TLS 1.3 and to provide the associated downgrade indications.

Tests
The evaluator shall follow the operational guidance as necessary to configure the TSF to negotiate only TLS 1.3 and to provide the associated downgrade indications. The evaluator shall send a TLS client hello to the TOE that indicates support for only TLS 1.2. The evaluator shall observe that the TSF sends a server hello with the last eight bytes of the server random value equal to 44 4F 57 4E 47 52 44 01.

FCS_TLSS_EXT.4 TLS Server Support for Renegotiation

This is a selection-based component. Its inclusion depends upon selection from FCS_TLS_EXT.1.1.

FCS_TLSS_EXT.4.1

The TSF shall support secure renegotiation through the use of [**selection**: *the "renegotiation_info" TLS extension, not allowing session renegotiation*] in accordance with RFC 5746.

FCS_TLSS_EXT.4.2

The TSF shall, when negotiating a TLS 1.2 session, [**selection**: *include the renegotiation_info extension in ServerHello messages when a client hello with the renegotiation_info extension is received and shall terminate a session if neither of the renegotiation_info or TLS_EMPTY_RENEGOTIATION_INFO_SCSV signaling ciphersuites are indicated in the client hello, not allow renegotiation*].

FCS_TLSS_EXT.4.3

The TSF shall terminate the session if an unexpected client hello is received during an active TLS session.

Application Note: RFC 5746 defines an extension to TLS 1.2 that binds renegotiation handshakes to the cryptography in the original handshake. As a refinement of the RFC, servers that support renegotiation and negotiating TLS 1.2 will terminate a session if neither of the methods described in RFC 5746 are offered by the client. RFC 5746 indicates that a server negotiating TLS 1.2 is required to terminate the session if the conditions for secure renegotiation are not met. Alternatively, a TLS server may negotiate TLS 1.2 without any RFC 5746 client renegotiation indicators if it always terminates an existing session when a new client hello is received, similar to the implementation of TLS 1.3.

TLS 1.3 does not allow renegotiation. Termination, as indicated in FCS_TLSS_EXT.4.3, covers TLS 1.3 sessions as well as TLS 1.2 sessions where the client hello received does not comply with RFC 5746, or when configured to reject renegotiation (if the product is configurable).

This SFR is claimed if "TLS as a server" is selected in FCS_TLS_EXT.1.1.

Evaluation Activities

FCS_TLSS_EXT.4
TSS
The evaluator shall examine the ST to confirm that the TLS description includes details on session renegotiation protection methods supported, to include when renegotiation is prohibited.

Guidance
The evaluator shall examine the operational guidance to confirm that any instructions that are needed to meet the requirements are included. If support for TLS 1.2 is configurable to use RFC 5746 methods or to deny renegotiation, the evaluator shall ensure that the operational guidance includes instructions for configuring the TSF in this manner.

Tests
The evaluator shall perform the following tests, as indicated for each version supported, using a test TLS client able to construct the indicated messages and expose messages received from the TSF:

- ***Test 20.1:*** *(RFC 5746 compliant TLS 1.2 initial handshake) [conditional] If the TSF supports renegotiation, the evaluator shall follow the operational guidance as necessary to configure the TSF to enforce RFC 5746 methods. The evaluator shall initiate a TLS 1.2 session from a test TLS client for each of the following sub-tests:*
  - ***Test 21.1.1:*** *The evaluator shall send an initial client hello without the renegotiation_info extension and without including the signaling ciphersuite value, TLS_EMPTY_RENEGOTIATION_INFO_SCSV. The evaluator shall observe that the TSF terminates the session.*

Note: *It is preferred that the TSF sends a fatal error alert message (e.g., handshake failure) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

- ***Test 21.1.2:*** *The evaluator shall send an initial client hello with the renegotiation_info extension indicating a renegotiated_connection length greater than zero. The evaluator shall observe that the TSF terminates the session.*

  Note: *It is preferred that the TSF sends a fatal error alert message (e.g., handshake failure) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

- ***Test 21.2:*** *(renegotiation attempt) For each of the following sub-tests, the evaluator shall establish a compliant TLS channel with an initial handshake that uses the indicated secure renegotiation method for the version indicated. Without closing the session, the evaluator shall send a second client hello within the channel specific to the version as indicated:*
  - ***Test 22.2.1:*** *[conditional] If the TSF allows renegotiation, the evaluator shall configure the TSF to support RFC 5746 methods, send an initial handshake with a valid renegotiation extension, send a new TLS 1.2 client hello on the TLS 1.2 channel containing the renegotiation_info extension indicating valid client_verify_data, and observe the TSF successfully completes the handshake.*
  - ***Test 22.2.2:*** *[conditional] If the TSF allows renegotiation, the evaluator shall send an initial client hello containing a valid renegotiation extension, send a new TLS 1.2 client hello on the TLS 1.2 channel with the signaling ciphersuite value, TLS_EMPTY_RENEGOTIATION_INFO_SCSV, and observe that the TSF terminates the session.*

    Note: *It is preferred that the TSF sends a fatal error alert message (e.g., handshake failure) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*
  - ***Test 22.2.3:*** *[conditional] If the TSF allows renegotiation, for each TLS 1.2 renegotiation method claimed in accordance with RFC 5746, the evaluator shall send an initial client hello indicating the method, send a new TLS 1.2 client hello on the TLS 1.2 channel without a renegotiation_info extension, and observe that the TSF terminates the session.*

    Note: *It is preferred that the TSF sends a fatal error alert message (e.g., unexpected message) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*
  - ***Test 22.2.4:*** *[conditional]: If the TSF allows renegotiation, for each TLS 1.2 renegotiation method claimed in accordance with RFC 5746, the evaluator shall send an initial client hello indicating the method, send a new TLS 1.2 client hello on the TLS 1.2 channel with a renegotiation_info extension indicating an invalid client_verify_data value (modify a byte of a valid value), and observe that the TSF terminates the session.*

    Note: *It is preferred that the TSF sends a fatal error alert message (e.g., unexpected message) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*
  - ***Test 22.2.5:*** *[conditional] If the TSF supports TLS 1.3, or if the TSF rejects renegotiation for TLS 1.2, then for each such version, the evaluator shall follow the operational guidance as necessary to configure the TSF to negotiate the version and reject renegotiation. The evaluator shall initiate a valid initial session for the specified version, send a valid client hello on the non-renegotiable TLS channel, and observe that the TSF terminates the session.*

    Note: *It is preferred that the TSF sends a fatal error alert message (e.g., unexpected message) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

FCS_TLSS_EXT.5 TLS Server Support for Session Resumption

This is a selection-based component. Its inclusion depends upon selection from FCS_TLSS_EXT.1.1.

FCS_TLSS_EXT.5.1

The TSF shall support session resumption as a server via the use of [**selection**: *session ID in accordance with RFC 5246*, *tickets in accordance with RFC 5077*, *PSK and tickets in accordance with RFC 8446*].

Application Note: The ST author indicates which session resumption mechanisms are supported. One or both of the first two options, "session ID in accordance with RFC 5246" and "tickets in accordance with RFC 5077" are

claimed for TLS 1.2 resumption. If resumption of TLS 1.3 sessions is supported, "PSK and tickets in accordance with RFC 8446" is selected, and the selection-based SFR FCS_TLSS_EXT.6 must also be claimed.

While it is possible to perform session resumption using PSK ciphersuites in TLS 1.2, this is uncommon. Validation of key exchange and session negotiation rules for PSK ciphersuites is independent of the source of the pre-shared key and is covered in FCS_TLSS_EXT.1.

This SFR is claimed if "session resumption" is selected in FCS_TLSS_EXT.1.1.

Evaluation Activities

FCS_TLSS_EXT.5
TSS
The evaluator shall examine the ST and confirm that the TLS server protocol description includes a description of the supported resumption mechanisms.

Guidance
The evaluator shall ensure the operational guidance describes instructions for any configurable features of the resumption mechanism.

Tests
The evaluator shall perform the following tests:

- ***Test 23.1:*** *For each supported version, and for each supported resumption method for that version, the evaluator shall establish a compliant initial TLS session with the TOE for the version using the specified method. The evaluator shall close the successful session and initiate resumption using the specified mechanism. The evaluator shall observe that the TSF successfully establishes the resumed session in accordance with the requirements.*
- ***Test 23.2:*** *For each supported version and each supported resumption method for the version, the evaluator shall send a compliant client hello message supporting only the specific version and indicating support for the resumption method. The evaluator shall allow the TOE and test client to continue with the compliant handshake until resumption information is established but then cause a fatal error to terminate the session. The evaluator shall then send a new client hello in an attempt to resume the session with the resumption information provided and verify that the TSF does not resume the session, but instead either terminates the session or completes a full handshake, ignoring the resumption information.*

  Note: *For TLS 1.2, resumption information should be established at the point the TSF sends a server hello, either acknowledging the session-based resumption or acknowledging support for ticket-based resumption and sending a new_ticket message. A TLS 1.2 session can then be terminated by sending a modified finished message. For TLS 1.3, the new_ticket message is sent after the finished message; once received by the client, the session can be terminated by modifying a byte of the encrypted application data.*

FCS_TLSS_EXT.6 TLS Server TLS 1.3 Resumption Refinements

This is a selection-based component. Its inclusion depends upon selection from FCS_TLSS_EXT.5.1.

FCS_TLSS_EXT.6.1
The TSF shall support TLS 1.3 resumption using PSK with psk_key_exchange_mode extension with the value psk_dhe.

FCS_TLSS_EXT.6.2
The TSF shall ignore early data received in TLS 1.3 sessions.

Application Note: This SFR is claimed when session resumption is supported for TLS 1.3. RFC 8446 allows pre-shared keys to be used directly and also allows early data to be protected using only the pre-shared key. This SFR refines the RFC to use PSK only with a supplemental DHE or ECDHE key exchange to ensure perfect forward secrecy for all sessions.

This SFR is claimed if "PSK and tickets in accordance with RFC 8446" is selected in FCS_TLSS_EXT.5.1.

Evaluation Activities

FCS_TLSS_EXT.6
TSS
The evaluator shall examine the ST to confirm that the TLS description includes details on session resumption for TLS 1.3, describes each application capable of using TLS 1.3 with PSK, and describes how the TSF and application respond to client attempts to use early data (including via logging or observable responses). The evaluator shall confirm that the TLS description shows that only the psk_dhe_ke psk_key_exchange_mode is supported and that early information is ignored.

Guidance
The evaluator shall examine the operational guidance to verify that instructions for any configurable features that are required to meet the requirement are included.

Tests
The evaluator shall follow the operational guidance to configure the TSF to negotiate TLS 1.3 and shall perform the following tests:

- ***Test 24.1:*** *The evaluator shall attempt a resumed session (as for FCS_TLSS_EXT.5 Test 1) but using psk_ke mode as the value for the psk_key_exchange_mode in the resumption client hello. The evaluator shall observe that the TSF refuses to resume the session, either by completing a full TLS 1.3 handshake or by terminating the session.*

  Note: *It is preferred that the TSF sends a fatal error alert message (e.g., illegal parameter) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

- ***Test 24.2:*** *The evaluator shall initiate a resumed session (as for FCS_TLSS_EXT.5 Test 1) with a test TLS 1.3 client attempting to provide early data that provokes a known reaction at the TOE if received. The evaluator shall observe that the TSF does not react to the early data, indicating that the data was ignored.*

  Note: *The specific early data used may depend on the applications calling the TLS session and should be selected to initiate an observable response in the TSF or calling application as described in the ST. For HTTPS, for example, the early data can be an HTTP POST that updates data at the TOE, which can then be observed via a user interface for the application if the data was posted or via application logging indicating that the operation failed.*

FCS_DTLSC_EXT.1 DTLS Client Protocol

This is a selection-based component. Its inclusion depends upon selection from FCS_TLS_EXT.1.1.

FCS_DTLSC_EXT.1.1

The product shall implement DTLS 1.2 (RFC 6347) and [**selection**: *DTLS 1.0 (RFC 4347), no earlier DTLS versions*] as a client that supports the ciphersuites [**selection**:

- *TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246*
- *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_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288*
- *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_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288*
- *TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289*
- *TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289*
- *TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289*
- *TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289*
- *TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289*
- *TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289*
- *TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289*
- *TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289*

] and also supports functionality for [**selection**:

- *mutual authentication*
- *none*

].

Application Note: If any ECDHE or DHE ciphersuites are selected, then FCS_TLSC_EXT.5 is required.

If *mutual authentication* is selected, then the ST must additionally include the requirements from FCS_DTLSC_EXT.2. If the TOE implements mutual authentication, this selection must be made.

Differences between DTLS 1.2 and TLS 1.2 are outlined in RFC 6347; otherwise the protocols are the same. All application notes listed for that are relevant to DTLS apply to this requirement.

FCS_DTLSC_EXT.1.2

The product shall verify that the presented identifier matches the reference identifier according to RFC 6125.

Application Note: All application notes listed for that are relevant to DTLS apply to this requirement.

FCS_DTLSC_EXT.1.3

The product shall not establish a trusted channel if the server certificate is invalid [**selection**: *with no exceptions*, *except when override is authorized*].

Application Note: All application notes listed for that are relevant to DTLS apply to this requirement.

FCS_DTLSC_EXT.1.4

The product shall [**selection, choose one of**: *terminate the DTLS session*, *silently discard the record*] if a message received contains an invalid MAC or if decryption fails in the case of GCM and other AEAD ciphersuites.

Evaluation Activities

FCS_DTLSC_EXT.1
Tests
The evaluator shall perform the evaluation activities listed for .

FCS_DTLSC_EXT.1.1
Tests
The evaluator shall perform the evaluation activities listed for , but ensuring that DTLS (and not TLS) is used in each evaluation activity.

For tests which involve version numbers, note that in DTLS the on-the-wire representation is the 1's complement of the corresponding textual DTLS version numbers. This is described in Section 4.1 of RFC 6347 and RFC 4347. For example, DTLS 1.0 is represented by the bytes 0xfe 0xff, while the undefined DTLS 1.4 would be represented by the bytes 0xfe 0xfb.

FCS_DTLSC_EXT.1.2
Tests
The evaluator shall perform the evaluation activities listed for .

FCS_DTLSC_EXT.1.4
TSS
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.

Tests
The evaluator shall establish a connection using a server. 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 DTLS Client Support for Mutual Authentication

This is a selection-based component. Its inclusion depends upon selection from FCS_DTLSC_EXT.1.1.

FCS_DTLSC_EXT.2.1

The product shall support mutual authentication using X.509v3 certificates.

Application Note: All application notes listed for FCS_TLSC_EXT.2.1 that are relevant to DTLS apply to this requirement.

Evaluation Activities

FCS_DTLSC_EXT.2

Tests
The evaluator shall perform the evaluation activities listed for FCS_TLSC_EXT.2.1.

FCS_DTLSS_EXT.1 DTLS Server Protocol

This is a selection-based component. Its inclusion depends upon selection from FCS_TLS_EXT.1.1.

FCS_DTLSS_EXT.1.1

The product shall implement DTLS 1.2 (RFC 6347) and [**selection**: *DTLS 1.0 (RFC 4347), no earlier DTLS versions*] as a server that supports the ciphersuites [**selection**:

- *TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246*
- *TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246*
- *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_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288*
- *TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288*
- *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_DHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288*
- *TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288*
- *TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289*
- *TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289*
- *TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289*
- *TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289*
- *TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289*
- *TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289*
- *TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289*
- *TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289*

] and no other ciphersuites, and also supports functionality for [**selection**:

- *mutual authentication*
- *none*

].

Application Note: If *mutual authentication* is selected, then the ST must additionally include the requirements from FCS_DTLSS_EXT.2. If the TOE implements mutual authentication, this selection must be made.

All application notes listed for FCS_TLSS_EXT.1.1 that are relevant to DTLS apply to this requirement.

FCS_DTLSS_EXT.1.2

The product shall deny connections from clients requesting [**assignment**: *list of DTLS protocol versions*].

Application Note: Any specific DTLS version not selected in FCS_DTLSS_EXT.1.1 should be assigned here. This version of the FP does not require the server to deny DTLS 1.0, and if the TOE supports DTLS 1.0 then "none" can be assigned. In a future version of this FP, DTLS 1.0 will be required to be denied.

FCS_DTLSS_EXT.1.3

The product shall not proceed with a connection handshake attempt if the DTLS Client fails validation.

Application Note: 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 1.0). The server validates the DTLS client during Connection Establishment (Handshaking) and prior to 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 a keyed hash function. 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 product shall perform key establishment for DTLS using [**selection**:

- *RSA with size [**selection**: 2048 bits, 3072 bits, 4096 bits, no other sizes]*

- *Diffie-Hellman parameters with size [**selection**: 2048 bits, 3072 bits, 4096 bits, 6144 bits, 8192 bits, no other size]*
- *Diffie-Hellman groups [**selection**: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, no other groups]*
- *ECDHE parameters using elliptic curves [**selection**: secp256r1, secp384r1, secp521r1] and no other curves*
- *no other key establishment methods*

].

Application Note: If the ST lists an RSA ciphersuite in FCS_DTLSS_EXT.1.1, the ST must include the RSA selection in the requirement.
If the ST lists a DHE ciphersuite in FCS_DTLSS_EXT.1.1, the ST must include either the Diffie-Hellman selection for parameters of a certain size, or for particular Diffie-Hellman groups.
If the ST lists an ECDHE ciphersuite in FCS_DTLSS_EXT.1.1, the ST must include the NIST curves selection in the requirement.

FCS_DTLSS_EXT.1.5

The product shall [**selection, choose one of**: *terminate the DTLS session*, *silently discard the record*] if a message received contains an invalid MAC or if decryption fails in the case of GCM and other AEAD ciphersuites.

Evaluation Activities

FCS_DTLSS_EXT.1.1
Tests
The evaluator shall perform the evaluation activities listed for FCS_TLSS_EXT.1.1, but ensuring that DTLS (and not TLS) is used in each stage of the evaluation activities.

For tests which involve version numbers, note that in DTLS the on-the-wire representation is the 1's complement of the corresponding textual DTLS version numbers. This is described in Section 4.1 of RFC 6347 and RFC 4347. For example, DTLS 1.0 is represented by the bytes 0xfe 0xff, while the undefined DTLS 1.4 would be represented by the bytes 0xfe 0xfb.

FCS_DTLSS_EXT.1.2
The following evaluation activities shall be conducted unless "none" is assigned.
TSS
The evaluator shall verify that the TSS contains a description of the denial of old DTLS versions consistent relative to selections in FCS_DTLSS_EXT.1.2.

Guidance
The evaluator shall verify that the operational guidance includes any configuration necessary to meet this requirement.

Tests
- ***Test 25.1:*** *The evaluator shall send a Client Hello requesting a connection with each version of DTLS specified in the selection and verify that the server denies the connection.*

FCS_DTLSS_EXT.1.3
TSS
The evaluator shall verify that the TSS describes how the DTLS Client IP address is validated prior to issuing a ServerHello message.

Tests
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
Tests
The evaluator shall perform the evaluation activities listed for .

FCS_DTLSS_EXT.1.5
TSS
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.

Tests
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 DTLS Server Support for Mutual Authentication

This is a selection-based component. Its inclusion depends upon selection from FCS_DTLSS_EXT.1.1.

FCS_DTLSS_EXT.2.1

The product shall support mutual authentication of DTLS clients using X.509v3 certificates.

Application Note: All application notes listed for that are relevant to DTLS apply to this requirement.

FCS_DTLSS_EXT.2.2

The product shall not establish a trusted channel if the client certificate is invalid.

Application Note: All application notes listed for that are relevant to DTLS apply to this requirement.

FCS_DTLSS_EXT.2.3

The product shall not establish a trusted channel if the Distinguished Name (DN) or Subject Alternative Name (SAN) contained in a certificate does not match one of the expected identifiers for the client.

Application Note: All application notes listed for that are relevant to DTLS apply to this requirement.

Evaluation Activities

FCS_DTLSS_EXT.2.1
Tests
The evaluator shall perform the evaluation activities listed for .

FCS_DTLSS_EXT.2.2
Tests
The evaluator shall perform the evaluation activities listed for .

FCS_DTLSS_EXT.2.3
Tests
The evaluator shall perform the evaluation activities listed for .

Appendix A - Implementation-based Requirements

Implementation-based Requirements are dependent on the TOE implementing a particular function. If the TOE fulfills any of these requirements, the vendor must either add the related SFR or disable the functionality for the evaluated configuration.

Appendix B - Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
Base-PP	Base Protection Profile
CA	Certificate Authority
CBC	Cipher Block Chaining
CC	Common Criteria
CEM	Common Evaluation Methodology
CN	Common Name
cPP	Collaborative Protection Profile
DHE	Diffie-Hellman Ephemeral
DN	Distinguished Name
DNS	Domain Name Server
DTLS	Datagram Transport Layer Security
EAP	Extensible Authentication Protocol
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
EP	Extended Package
FP	Functional Package
GCM	Galois/Counter Mode
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
NIST	National Institute of Standards and Technology
OE	Operational Environment
PP	Protection Profile
PP-Configuration	Protection Profile Configuration
PP-Module	Protection Profile Module
RFC	Request for Comment (IETF)
RSA	Rivest Shamir Adelman
SAN	Subject Alternative Name
SAR	Security Assurance Requirement
SCSV	Signaling ciphersuite Value
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
ST	Security Target
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TOE	Target of Evaluation

TSF	TOE Security Functionality
TSFI	TSF Interface
TSS	TOE Summary Specification
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

Appendix C - Bibliography

Identifier	Title
[CC]	Common Criteria for Information Technology Security Evaluation - • Part 1: Introduction and General Model, CCMB-2017-04-001, Version 3.1 Revision 5, April 2017. • Part 2: Security Functional Components, CCMB-2017-04-002, Version 3.1 Revision 5, April 2017. • Part 3: Security Assurance Components, CCMB-2017-04-003, Version 3.1 Revision 5, April 2017.