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.

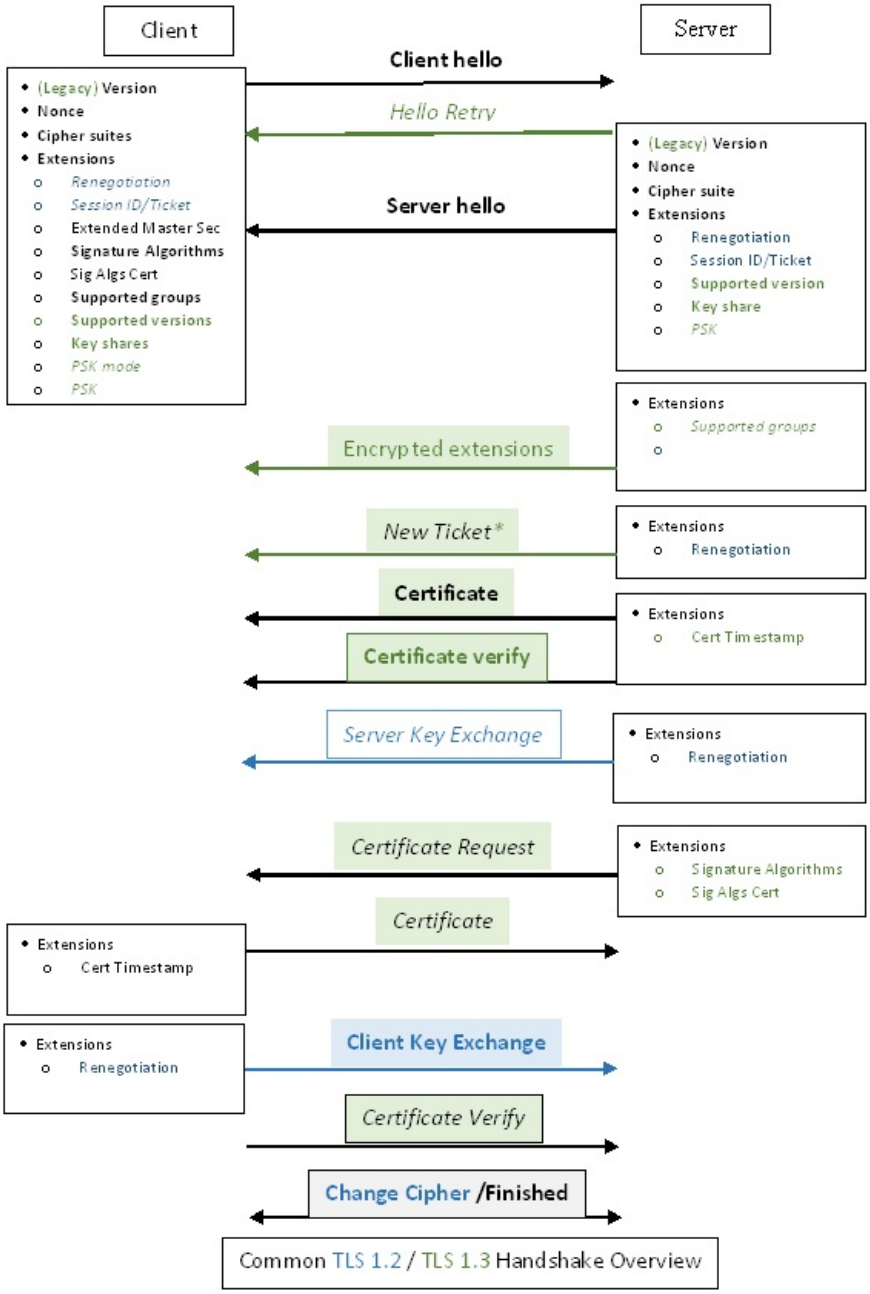

Figure 1: TLS Hello

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.

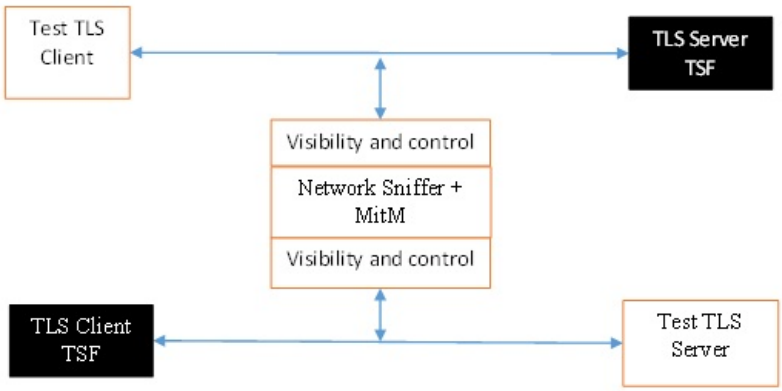

Figure 2: Test environment for TLS 1.2 using network traffic visibility and control tools

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.

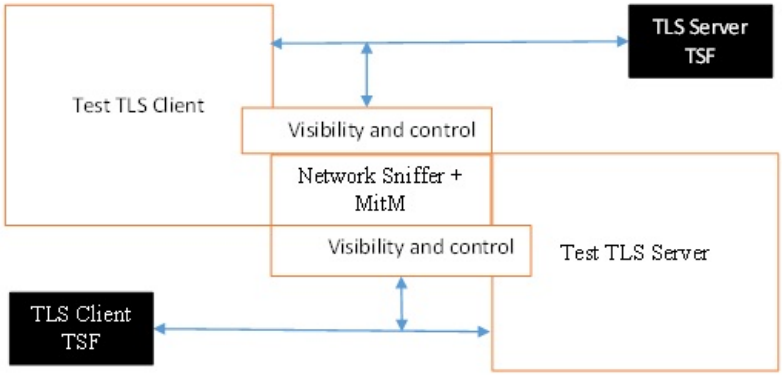

Figure 4: Combined test environment for TLS 1.2 and TLS 1.3 using both network tools and custom endpoint capabilities

FCS_TLSC_EXT.1 TLS Client Protocol

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

FCS_TLSC_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 client that supports additional functionality for session renegotiation protection and [**selection**:

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

] and shall abort attempts by a server to negotiate all other TLS or SSL versions.

Application Note: Session renegotiation protection is required for both TLS 1.2 and TLS 1.3, and the ST must include the requirements from FCS_TLSC_EXT.4. Within FCS_TLSC_EXT.4, options for implementation of secure session renegotiation for TLS 1.2, or rejecting renegotiation requests are claimed.

The ST author will claim TLS 1.3 functionality if supported, and optional functionality as appropriate for the claimed versions.

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

If "supplemental downgrade protection" is selected, then the ST must additionally include the requirements from FCS_TLSC_EXT.3. This is claimed if TLS 1.3 is supported, or if the product supports TLS 1.1 or below downgrade protection using the mechanism described in RFC 8446.

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

FCS_TLSC_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]]*

  ]

] offering the supported ciphersuites in a client hello message in preference order: [**assignment**: *list of supported ciphersuites*].

Application Note: The ST author should select the ciphersuites that are supported, and must select at least one ciphersuite for each TLS version supported. The ciphersuites to be tested in the evaluated configuration are limited by this requirement. However, this requirement does not restrict the TOE's ability to propose additional non-deprecated ciphersuites beyond the ones listed in this requirement in its Client Hello message as indicated in the ST. That is, the TOE may propose any ciphersuite not excluded by this element, but the evaluation will only test ciphersuites from the above list. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment.

TLS 1.3 ciphersuites are claimed if support for TLS 1.3 is claimed in FCS_TLSC_EXT.1.1. The assignment of preference order provides an ordered list of all supported ciphersuites with the most preferred ciphersuites listed first. Ciphersuites listed in [RFC 9151, “CNSA Suite TLS Profile”] are preferred over all other ciphersuites, GCM ciphersuites are preferred over CBC ciphersuites, ECDHE preferred over RSA and DHE, and SHA256 or SHA384 over SHA1.

Ciphersuites for TLS 1.2 are of the form TLS_{key exchange algorithm}_WITH_{encryption algorithm}_(message digest algorithm}, and are listed in the TLS parameters section of the internet assignments at iana.org.

FCS_TLSC_EXT.1.3

The TSF shall not offer the following ciphersuites indicating the following:

- the null encryption component
- support for anonymous servers
- use of deprecated or export-grade cryptography including DES, 3DES, RC2, RC4, or IDEA for encryption
- use of MD

and shall abort sessions where a server attempts to negotiate ciphersuites not enumerated in the client hello message.

FCS_TLSC_EXT.1.4

The TSF shall be able to support 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 server 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)*
    - *post_handshake_auth (RFC 8446), pre_shared_key (RFC 8446), and psk_key_exchange_mode (RFC 8446) indicating DHE or ECDHE mode*
    - *no other extensions*

  ] and shall not send the following extensions:
    - early_data
    - psk_key_exchange_mode indicating PSK only mode.

Application Note: If TLS 1.3 is claimed in FCS_TLSC_EXT.1.1, supported_versions, supported_groups, and key_share extensions are claimed in accordance with RFC 8446. If TLS 1.3 is not claimed, supported_versions and key_share extensions are not claimed. Other extensions may be supported; certain extensions may need to be claimed based on other SFR claims made.

If ECDHE ciphersuites are claimed in FCS_TLSC_EXT.1.2, the supported_groups extension is claimed here with appropriate secp groups claimed. If DHE ciphersuites are claimed in FCS_TLSC_EXT.1.2, it is preferred that the appropriate ffdhe groups be claimed here. In a subsequent version of this FP, support for ffdhe groups will be required whenever DHE ciphersuites are claimed.

When 'other non-deprecated signature algorithms' is claimed, the assignment will describe the standard signature and hash algorithms supported. MD5 and SHA-1 hashes are deprecated and are not included in the signature_algorithms or signature_algorithms_cert extensions.

FCS_TLSC_EXT.1.5

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

- *verify that a presented identifier of name type: [**selection**:*
  - *DNS name type according to RFC 6125*
  - *URN name type according to RFC 6125*
  - *SRV name type according to RFC 6125*
  - *Common Name conversion to DNS name according to RFC 6125*
  - *Directory name type according to RFC 5280*
  - *IPaddress name type according to RFC 5280*
  - *rfc822Name type according to RFC 5280*
  - *[**assignment**: other name type]*

  ]
- *interface with a client application requesting the TLS channel to verify that a presented identifier*

] matches a reference identifier of the requested TLS server and shall abort the session if no match is found.

Application Note: The rules for verification identity are described in Section 6 of RFC 6125 and Section 7 of RFC 5280. The reference identifier is established by the user (e.g., entering a URL into a web browser or clicking a link), by configuration (e.g., configuring the name of a mail server or authentication server), or by an application (e.g., a parameter of an API) depending on the product service. The client establishes all acceptable reference identifiers and interfaces with the TLS implementation to provide acceptable reference identifiers, or to accept the presented identifiers as validated in the server's certificate. If the product performs matching of the reference identifiers to the identifiers provided in the server's certificate, the first option is claimed and all supported name types are claimed; if the product presents the certificate, or the presented identifiers from the certificate to the application, the second option is claimed.

In most cases where TLS servers are represented by DNS-type names, the preferred method for verification is the Subject Alternative Name using DNS names, URI names, or Service Names. Verification using a conversion of the Common Name relative distinguished name from a DNS name type in the subject field is allowed for the purposes of backward compatibility.

Finally, the client should avoid constructing reference identifiers using wildcards. However, if the presented identifiers include wildcards, the client must follow the best practices regarding matching; these best practices are captured in the evaluation activity. Support for other name types is rare, but may be claimed for specific applications.

[JF] App note talks about wildcards best practices being captured in the evaluation activity but nothing in the activity actually covers this.

FCS_TLSC_EXT.1.6

The TSF shall not establish a trusted channel if the server certificate is invalid [**selection**: *with no exceptions*, *except when override is authorized in the case where valid revocation information is not available*].

Application Note: A certificate used in a manner that does not support revocation checking should not advertise revocation information locations. Common methods to address this include revoking the issuing CA, resetting certificate pinning mechanisms, or removing entries from trust stores. Thus, a certificate that does not advertise revocation status information is considered to be not revoked and does not need to be processed via override mechanisms. Override mechanisms are for use with certificates with published revocation status information that is not accessible, whether temporarily or because the information cannot be accessed during the state of the TOE (e.g., for verifying signatures on boot code). The circumstances should be described by the ST author, who should indicate the override mechanism and conditions that apply to the override, including system state, user/admin actions, etc.

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

Evaluation Activities

FCS_TLSC_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) and as refined in FCS_TLSC_EXT.1 as appropriate.

The evaluator shall verify that ciphersuites indicated in FCS_TLSC_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_TLSC_EXT.1.4.

The evaluator shall verify that the ST describes applications that use the TLS functions and how they establish reference identifiers.

The evaluator shall verify that the ST includes a description of the name types parsed and matching methods supported for associating the server certificate to application defined reference identifiers.

Guidance
The evaluator shall check the operational guidance to ensure that it contains instructions on configuring the product so that TLS conforms to the description in the TSS and that it includes any instructions on configuring the version, ciphersuites, or optional extensions that are supported.

The evaluator shall verify that all configurable features for matching identifiers in certificates presented in the TLS handshake to application specific reference identifiers are described.

Tests
The evaluator shall perform the following tests:

- ***Test 1.1:*** *(supported configurations) For each supported version, and for each supported ciphersuite associated with the version:*

  The evaluator shall establish a TLS connection between the TOE and a test TLS server that is configured to negotiate the tested version and ciphersuite in accordance with the RFC for the version.

  The evaluator shall observe that the TSF presents a client hello with the highest version of TLS 1.2 or the legacy version (value '03 03') and shall observe that the supported version extension is not included for TLS 1.2, and, if TLS 1.3 is supported, is present and contains the value '03 04' for TLS 1.3.

  The evaluator shall observe that the client hello indicates the supported ciphersuites in the order indicated, and that it includes only the extensions supported, with appropriate values, for that version in accordance with the requirement.

  The evaluator shall observe that the TOE successfully completes the TLS handshake.

  Note: *TOEs supporting TLS 1.3, but allowing a server to negotiate TLS 1.2, should include all ciphersuites and all extensions as required for either version. If such a TOE is configurable to support only TLS 1.2, only TLS 1.3, or both TLS 1.2 and TLS 1.3, Test 1*

should be performed in each configuration – with advertised ciphersuites appropriate for the configuration.

The connection 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).

- ***Test 1.2:*** *(obsolete versions) The evaluator shall perform the following tests:*
  - ***Test 2.2.1:*** *For each of SSL version 3, TLS version 1.0, and TLS version 1.1, the evaluator shall initiate a TLS connection from the TOE to a test TLS server that is configured to negotiate the obsolete version and observe that 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 2.2.2:*** *The evaluator shall attempt to establish a connection with a test TLS server that is configured to send a server hello message indicating the selected version (referred to as the legacy version for TLS 1.3) with a value corresponding to an undefined TLS (legacy) version (e.g., '03 04') and observe that the TSF terminates the connection.*

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

    Test 2.2 is intended to test the TSF response to non-standard versions, including early proposals for 'beta TLS 1.3' versions. RFC 8446 requires the legacy version to have the value '03 03' and specifies TLS 1.3 in the supported versions extension with the value '03 04'. While not a preferred approach, if continued support for a beta TLS 1.3 version is desired and the TSF cannot be configured to reject such versions, another value (e.g., '03 05') can be used in Test 2.2. Implementations of non-standard versions are not tested.
- ***Test 2.3:*** *(ciphersuites) The evaluator shall perform the following tests on handling unexpected ciphersuites using a test TLS server sending handshake messages compliant with the negotiated version except as indicated in the test:*
  - ***Test 3.3.1:*** *(ciphersuite not offered) For each supported version, the evaluator shall attempt to establish a connection with a test TLS server configured to negotiate the supported version and a ciphersuite not included in the client hello 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) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

    This test intended to test the TSF's generic ability to recognize non-offered ciphersuites. If the ciphersuites in the client hello are configurable, the evaluator shall configure the TSF to offer a ciphersuite outside those that are supported and use that ciphersuite in the test. If the TSF ciphersuite list is 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 3.3.2:*** *(version confusion) For each supported version, the evaluator shall attempt to establish a connection with a test TLS server that is configured to negotiate the supported version and a ciphersuite that is not associated with that version 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) 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 intended that Test 3.2 use TLS 1.3 ciphersuites for a server negotiating TLS 1.2. If TLS 1.3 is supported, the test server negotiating TLS 1.3 should select a TLS 1.2 ciphersuite supported by the TOE for TLS 1.2 and matching the client's supported groups and signature algorithm indicated by extensions in the TLS 1.3 client hello. If the TOE is configurable to allow both TLS 1.2 and TLS 1.3 servers, the test server should use ciphersuites offered by the TSF in its client hello message.
  - ***Test 3.3.3:*** *(null ciphersuite) For each supported version, the evaluator shall attempt to establish a connection with a test TLS server configured to negotiate the null ciphersuite (TLS_NULL_WITH_NULL_NULL) 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 3.3.4:*** *(anon ciphersuite) The evaluator shall attempt to establish a TLS 1.2 connection with a test TLS server configured to negotiate a ciphersuite using the anonymous server authentication method and observe that the TOE rejects the connection.*

  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 selected by the test TLS server. The test ciphersuite should use supported cryptographic algorithms for as many of the other components as possible. For example, if the TSF only supports the ciphersuite TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, the test server could select TLS_DH_ANON_WITH_AES_256_GCM_SHA_384.
- ***Test 3.3.5:*** *(deprecated encryption algorithm) For each deprecated encryption algorithm (NULL, RC2, RC4, DES, IDEA, and TDES), the evaluator shall attempt to establish a TLS 1.2 connection with a test TLS server configured to negotiate a ciphersuite using the deprecated encryption algorithm 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 tested. The test ciphersuite should use supported cryptographic algorithms for as many of the other components as possible. For example, if the TSF only supports TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, the test server could select 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 3.4:*** *(extensions) For each supported version indicated in the following tests, the evaluator shall establish a connection from the TOE with a test server negotiating the tested version and providing server handshake messages as indicated when performing the following tests for validating proper extension handling:*
  - ***Test 4.4.1:*** *(signature_algorithms) [conditional] If the TSF supports certificate-based server authentication, the evaluator shall perform the following tests:*
    - ***Test 5.4.1.1:*** *For each supported version, the evaluator shall initiate a TLS session with a TLS test server and observe that the TSF's client hello includes the signature_algorithms extension with values in conformance with the ST.*
    - ***Test 5.4.1.2:*** *(TLS 1.2 only) [conditional] If the TSF supports an ECDHE or DHE ciphersuite, the evaluator shall ensure the test TLS server sends a compliant server hello message selecting TLS 1.2 and one of the supported ECDHE or DHE ciphersuites, a compliant server certificate message, and a key exchange message signed using a signature algorithm and hash combination not included in the client's hello message (e.g., RSA with SHA-1). 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, decryption 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 5.4.1.3:*** *[conditional] If TLS 1.3 is supported, the evaluator shall configure the test TLS server to respond to the TOE with a compliant server hello message selecting TLS 1.3 and a server certificate message, but then also sends a certificate verification message that uses a signature algorithm method not included in the signature_algorithms extension. The evaluator shall observe that the TSF terminates the TLS handshake.*

      Note: *It is preferred that the TSF sends a fatal error alert message (e.g., handshake failure, illegal parameter, bad certificate, decryption 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 5.4.1.4:*** *[conditional] For all supported versions for which signature_algorithms_cert is not supported, the evaluator shall ensure the test TLS server sends a compliant server hello message for the tested version and a server certificate message containing a valid certificate that represents the test TLS server, but which is signed using a signature and hash combination not included in the TSF's signature_algorithms extension (e.g., a certificate signed using RSA and SHA-1). The evaluator shall observe that the TSF terminates the TLS session.*

Note: *It is preferred that the TSF sends a fatal error alert message (e.g., unsupported certificate, bad certificate, decryption error, 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).*

Certificate-based server authentication is required unless the TSF only supports TLS with shared PSK. For TLS 1.2, this is the case if only 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, or TLS_DHE_PSK_WITH_AES_128_GCM_SHA256 as defined in RFC 5487, are supported. For TLS 1.3, this is the case if only PSK handshakes are supported.

- ***Test 5.4.2:*** *(signature_algorithms_cert) [conditional] If signature_algorithms_cert is supported, then for each version that uses the signature_algorithms_cert extension, the evaluator shall ensure that the test TLS server sends a compliant server hello message selecting the tested version and indicating certificate-based server authentication.*

  The evaluator shall ensure that the test TLS server forwards a certificate message containing a valid certificate that represents the test TLS server, but which is signed by a valid Certification Authority using a signature and hash combination not included in the TSF's signature_algorithms_cert extension (e.g., a certificate signed using RSA and SHA-1). The evaluator shall confirm the TSF terminates the session.

  Note: *Support for certificate based authentication is assumed if the signature_algorithms_cert is supported. For TLS 1.2, a non-PSK ciphersuite, or one of TLS_RSA_PSK_WITH_AES_256_GCM_SHA384 or TLS_RSA_PSK_WITH_AES_128_GCM_SHA256 as defined in RFC 5487 is used to indicate certificate-based server authentication. For TLS 1.3, the test server completes a full handshake, even if a PSK is offered to indicate certificate-based server authentication. If the TSF only supports shared PSK authentication, Test 4.2 is not performed.*

  For TLS 1.3, the server certificate message is encrypted. The evaluator will configure the test TLS server with the indicated certificate and ensure that the certificate is indeed sent by observing the buffer of messages to be encrypted, or by inspecting one or both sets of logs from the TSF and test TLS server.

  It is preferred that the TSF sends a fatal error alert message (e.g., unsupported certificate, bad certificate, decryption error, 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 5.4.3:*** *(extended_master_secret) (TLS 1.2 only) The evaluator shall initiate a TLS 1.2 session with a test TLS server configured to compute a master secret according to RFC 5246, section 8.*

  The evaluator shall observe that the TSF's client hello includes the extended master secret extension in accordance with RFC 7627, and ensures that the test TLS server does not include the extended master secret extension in its server hello. 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 5.4.4:*** *(supported_groups) (TLS 1.2 only – for TLS 1.3, testing is combined with testing of the keyshare extension)*
  - ***Test 6.4.4.1:*** *For each supported group, the evaluator shall initiate a TLS session with a compliant test TLS 1.2 server supporting RFC 7919. The evaluator shall ensure that the test TLS server is configured to select TLS 1.2 and a ciphersuite using the supported group. The evaluator shall observe that the TSF's client hello lists the supported groups as indicated in the ST, and that the TSF successfully establishes the TLS session.*
  - ***Test 6.4.4.2:*** *[conditional on TLS 1.2 support for ECDHE ciphersuites] The evaluator shall initiate a TLS session with a test TLS server that is configured to use an explicit version of a named EC group supported by the client. The evaluator shall ensure that the test TLS server key exchange message includes the explicit formulation of the group in its key exchange message as indicated in RFC 4492 section 5.4. 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., 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 6.5:*** *(TLS 1.3 extensions) [conditional] If the TSF supports TLS 1.3, the evaluator shall perform the following tests. For each test, the evaluator shall observe that the TSF's client*

hello includes the supported versions extension with the value '03 04' indicating TLS 1.3:

- ***Test 7.5.1:*** *(supported versions) The evaluator shall initiate TLS 1.3 sessions in turn from the TOE to a test TLS server configured as indicated in the sub-tests below:*
  - ***Test 8.5.1.1:*** *The evaluator shall configure the test TLS server to include the supported versions extension in the server hello containing the value '03 03.' The evaluator shall observe that the TSF terminates the TLS session.*

    Note: *It is preferred that the TSF sends a fatal error alert message (e.g., illegal parameter, handshake failure, protocol version) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*
  - ***Test 8.5.1.2:*** *The evaluator shall configure the test TLS server to include the supported versions extension in the server hello containing the value '03 04' and complete a compliant TLS 1.3 handshake. The evaluator shall observe that the TSF completes the TLS 1.3 handshake successfully.*
  - ***Test 8.5.1.3:*** *[conditional] If the TSF is configurable to support both TLS 1.2 and TLS 1.3, the evaluator shall follow operational guidance to configure this behavior. The evaluator shall ensure that the test TLS server sends a TLS 1.2 compliant server handshake and observe that the server random does not incidentally include any downgrade messaging. The evaluator shall observe that the TSF completes the TLS 1.2 handshake successfully.*

    Note: *Enhanced downgrade protection defined in RFC 8446 is optional, and if supported, is tested separately. The evaluator may configure the test server's random, or may repeat the test until the server's random does not match a downgrade indicator.*
- ***Test 8.5.2:*** *(supported groups, key shares) The evaluator shall initiate TLS 1.3 sessions in turn with a test TLS server configured as indicated in the following sub-tests:*
  - ***Test 9.5.2.1:*** *For each supported group, the evaluator shall configure the compliant test TLS 1.3 server to select a ciphersuite using the group. The evaluator shall observe that the TSF sends an element of the group in its client hello key shares extension (after a hello retry message from the test server, if the key share for the group is not included in the initial client hello). The evaluator shall ensure the test TLS server sends an element of the group in its server hello and observes that the TSF completes the TLS handshake successfully.*
  - ***Test 9.5.2.2:*** *For each supported group, the evaluator shall modify the server hello sent by the test TLS server to include an invalid key share value claiming to be an element the group indicated in the supported groups extension. The evaluator shall observe that the TSF terminates the TLS 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).*

    For DHE ciphersuites, a zero value, or a value greater or equal to the modulus is not a valid element. For ECDHE groups, an invalid point contains x and y coordinates of the correct size, but represents a point not on the curve. The evaluator can construct such an invalid point by modifying a byte in the y coordinate of a valid point and verify that the coordinates do not satisfy the curve equation.
- ***Test 9.5.3:*** *(PSK support) [conditional] If the TOE supports pre-shared keys, the evaluator shall follow the operational guidance to use pre-shared keys, shall establish a pre-shared key between the TSF and the test TLS server, and initiate TLS 1.3 sessions in turn between the TSF and the test TLS server configured as indicated in the following sub-tests:*
  - ***Test 10.5.3.1:*** *The evaluator shall configure the TSF to use the pre-shared key and ensure that the test TLS server functions as a compliant TLS 1.3 server. The evaluator shall observe that the TSF's client hello includes the pre_shared_key extension with the valid PSK indicator shared with the test server. The evaluator shall also observe that the TSF's client hello also includes the psk_key_exchange_mode and the post_handshake_auth extensions and that the psk_key_exchange_mode indicates one or more of DHE or ECDHE modes but does not include the PSK-only mode. The evaluator shall observe that the TSF completes the TLS 1.3 handshake successfully in accordance with RFC 8446, to include the TSF sending appropriate key shares for one or more of the supported groups.*

    Once the handshake is successful, the evaluator shall cause the test TLS server to send a certificate request and observe that the TSF provides a certificate message and certificate verify message.

    Note: *It may be necessary to complete a standard handshake and send a new-*

ticket message from the test TLS server to establish a pre-shared key, or it might be possible to configure the pre-shared key manually via out-of-band mechanisms. This can be performed in conjunction with other testing that is not tested as part of this SFR. It is not required at this time to support emerging standards on establishing PSK, but as such standards are finalized, this FP may be updated to require such support.

TLS messages after the handshake are encrypted so it may not be possible to observe the certificate and certificate verify messages sent by the TSF directly. The evaluator may need to configure the test TLS server to use an application that requires post-handshake client authentication and terminates the session or otherwise has an observable effect if the certificate is not provided.

- ***Test 10.5.3.2:*** *The evaluator shall attempt to configure the TSF to send early data. If there is no indication from the TSF that this is blocked, the evaluator shall repeat test 5.3.1 with the TSF so configured and observe that the TSF does not send application data prior to receiving the server hello.*

  Note: *Early data will be encrypted under the PSK and received by the test TLS server prior to it sending a server hello message.*

- ***Test 10.6:*** *(corrupt finished message) For each supported version, the evaluator shall initiate a TLS session from the TOE to a test TLS server that sends a compliant set of server handshake messages, except for sending a modified finished message (modify a byte of the finished message that would have been sent by a compliant server). The evaluator shall observe that the TSF terminates the session and does not complete the handshake by observing that the TSF does not send application data provided to the TLS channel.*
- ***Test 10.7:*** *(missing finished message) For each supported version, the evaluator shall initiate a session from the TOE to a test TLS server providing a compliant handshake, except for sending a random TLS message (the five byte header indicates a correct TLS message for the negotiated version, but not indicating a finished message) as the final message. The evaluator shall observe that the TSF terminates the session and does not send application data.*

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

  For TLS 1.2, the modified message is sent after the change_cipher_spec message. For TLS 1.3, the modified message is sent as the last message of the server's second flight of messages.

- ***Test 10.8:*** *(unexpected/corrupt signatures within handshake) The evaluator shall perform the following tests, according to the versions supported.*
  - ***Test 11.8.1:*** *(TLS 1.2 only) [conditional] If the ST indicates support for ECDSA or DSA ciphersuites, the evaluator shall initiate a TLS session with a compliant test TLS server and modify the signature in the server key exchange. The evaluator shall observe that the TSF terminates the session with a fatal alert message (e.g., decrypt error, handshake error).*
  - ***Test 11.8.2:*** *[conditional] If the ST indicates support for TLS 1.3, the evaluator shall initiate a TLS session between the TOE and a test TLS server that is configured to send a compliant server hello message, encrypted extension message, and certificate message, but will send a certificate verify message with an invalid signature (e.g., by modifying a byte from a valid signature). The evaluator shall confirm that the TSF terminates the session with a fatal error alert message (e.g., bad certificate, decrypt error, handshake error).*
  - ***Test 11.8.3:*** *(TLS 1.2 only) [conditional] If the ST indicates support for both RSA and ECDSA methods in the signature_algorithm (or, if supported, the signature_algorithms_cert) extension, and if the ST indicates one or more TLS 1.2 ciphersuites indicating each of the RSA and ECDSA methods in its signature components, the evaluator shall choose two ciphersuites: one indicating an RSA signature (cipher 1) and one indictaing an ECDSA signature (cipher 2). The evalutor shall then establish establish two certificates that are trusted by the TOE: one representing the test TLS 1.2 server using an RSA signature (cert 1) and one representing the test TLS 1.2 server using an ECDSA signature (cert 2). The evaluator shall initiate a TLS session between the TOE and the test TLS 1.2 server that is configured to select cipher 1 and to send cert 2. The evaluator shall verify that the TSF terminates this TLS session. The evaluator shall then initiate a TLS session between the TOE and the test 1.2 server that is configured to select cipher 2 and to send cert 1. The evaluator shall verify that the TSF also terminates this TLS session.*

    Note: *It is preferred that the TSF sends a fatal error alert message (e.g., bad certificate, decryption error, 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 11.9:*** *[conditional] If the TSF supports certificate-based server authentication, then*

for each supported version, the evaluator will initiate a TLS session from the TOE to the compliant test TLS server configured to negotiate the tested version, and to authenticate using a certificate trusted by the TSF as specified in the following:

- ***Test 12.9.1:*** *(certificate extended key usage purpose) The evaluator shall send a server certificate that contains the Server Authentication purpose in the extendedKeyUsage extension and verify that a connection is established. The evaluator shall repeat this test using a different certificate that is otherwise valid and trusted but lacks the Server Authentication purpose in the extendedKeyUsage extension and observe the TSS terminates the session.*

  Note: This test may be performed as part of certificate validation testing (FIA_X509_EXT.1).

  It is preferred that the TSF sends a fatal error alert message (e.g., bad certificate, decryption error, 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).

  Ideally, the two certificates should be similar in regards to structure, the types of identifiers used, and the chain of trust.

- ***Test 12.9.2:*** *(certificate identifiers) For each supported method of matching presented identifiers, and for each name type for which the TSF parses the presented identifiers from the server certificate for the method, the evaluator shall establish a valid certificate trusted by the TSF to represent the test server using only the tested name type. The evaluator shall perform the following sub-tests:*
  - ***Test 13.9.2.1:*** *The evaluator shall prepare the TSF as necessary to use the matching method and establish reference identifiers for the test server for the tested name type. The evaluator shall ensure the test TLS server sends a certificate with a matching name of the tested name type and observe that the TSF completes the connection.*
  - ***Test 13.9.2.2:*** *The evaluator shall prepare the TSF as necessary to use the matching method and establish reference identifiers that do not match the name representing the test server. The evaluator shall ensure the test TLS server sends a certificate with a name of the type tested, and observe the TSF terminates the session.*

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

- ***Test 13.9.3:*** *(mixed identifiers)[conditional] If the TSF supports a name matching method where the TSF performs matching of both CN-encoded name types and SAN names of the same type, then for each such method, and for eah such name type, the evaluator shall establish a valid certificate trusted by the TSF to represent the test server using one name for the CN-encoded name type and a different name for the SAN name type The evaluator shall perform the following tests:*
  - ***Test 14.9.3.1:*** *The evaluator shall follow the operational guidance to configure the TSF to use the name matching method and establish reference identifiers matching only the SAN. The evaluator shall ensure that the test server sends the certificate with the matching SAN and non-matching CN-encoded name, and observe that the TSF completes the connection.*

    Note: *Configuration of the TSF may depend on the application using TLS.*
  - ***Test 14.9.3.2:*** *The evaluator shall follow the operational guidance to configure the TSF to use the name matching method and establish reference identifiers matching only the CN-encoded name. The evaluator shall ensure that the test server sends the certificate with the matching SAN name and non-matching CN-encoded name, and observe that the TSF terminates the session.*

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

- ***Test 14.9.4:*** *(empty certificate) The evaluator shall configure the test TLS server to supply an empty certificate message and verify 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) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).*

- ***Test 14.9.5:*** *(invalid certificate) [conditional] If validity exceptions are supported, then for each exception for certificate validity supported, the evaluator shall configure the TSF to allow the exception and ensure the test TLS server sends a certificate that is valid and trusted, except for the allowed exception. The evaluator shall observe that the TSF completes the session.*

  Without modifying the TSF configuration, the evaluator shall initiate a new session with the test TLS server that includes an additional validation error, and observe that

the TSF terminates the session.

***Note:** It is preferred that the TSF sends a fatal error alert message (e.g., decode error, bad certificate) 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 verify the scope of the exception processing. If verifying certificate status information is claimed as an exception, then this test will verify that a TLS session succeeds when all supported methods for obtaining certificate status information is blocked from the TSF, to include removing any status information that might be cached by the TSF. If the exception is limited to specific certificates (e.g., only leaf certificates are exempt, or only certain leaf certificates are exempt) the additional validation error could be unavailable revocation information for a non-exempt certificate (e.g., revocation status information from an intermediate CA is blocked for the issuing CA of an exempt leaf certificate, or revocation information from the issuing CA is blocked for a non-exempt leaf certificate). If the only option for the exception is for all revocation information for all certificates, another validation error from FIA_X509_EXT.1 (e.g., certificate expiration, extended key usage, etc.) may be used.

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 FCS_TLSC_EXT.1.1 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 FCS_TLSC_EXT.1.1, 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 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_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 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 , 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 15.1:*** *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.