Requirements from the

Functional Package for Transport Layer Security (TLS)

Version: 1.1

2019-03-01

National Information Assurance Partnership

Revision History

Version	Date	Comment

Introduction

Purpose. This document presents the functional and assurance requirements found in the *Functional Package for Transport Layer Security (TLS)*. Common Criteria evaluation, facilitated in the U.S. by the National Information Assurance Partnership (NIAP), is required for IA and IA-enabled products in National Security Systems according to CNSS Policy #11.

Using this document. This representation of the Protection Profile includes:

- *Security Functional Requirements* for use in evaluation. These are featured without the formal Assurance Activities specified in the Protection Profile, to assist the reader who is interested only in the requirements.

  It also includes, in tables shown later, particular types of security functional requirements that are not strictly required in all cases. These are:

  - *Selection-based Security Functional Requirements* which become required when certain selections are made inside the regular Security Functionality Requirements (as indicated by the **[selection:]** construct).
  - *Objective Security Functional Requirements* which are highly desired but not yet widely-available in commercial technology.
  - *Optional Security Functional Requirements* which are available for evaluation and which some customers may insist upon.

- *Security Assurance Requirements* which relate to developer support for the product under evaluation, development processes, and other non-functionality security relevant requirements.

Security Functional Requirements

TLS Protocol

FCS_TLS_EXT.1.1 The product 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.

Security Assurance Requirements

Selection-Based Security Functional Requirements

TLS Client Protocol

FCS_TLSC_EXT.1.1 The product shall implement TLS 1.2 (RFC 5246) and [**selection**: *TLS 1.1 (RFC 4346)*, *no earlier TLS versions*] as a client that supports the cipher suites [**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*,
- *session renegotiation*,
- *none*

].

Application Note: The ST author should select the cipher suites that are supported, and must select at least one cipher suite. The cipher suites 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 cipher suites beyond the ones listed in this requirement in its Client Hello message. That is, the TOE may propose any cipher suite but the evaluation will only test cipher suites from the above list. It is necessary to limit the cipher suites that can be used in an evaluated configuration administratively on the server in the test environment. GCM cipher suites are preferred over CBC cipher suites, ECDHE preferred over RSA and DHE, and SHA256 or SHA384 over SHA.

TLS_RSA_WITH_AES_128_CBC_SHA is not required despite being mandated by RFC 5246.

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

If any ECDHE or DHE cipher suites are selected, then FCS_TLSC_EXT.5 is required.

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 *session renegotiation* is selected, then the ST must additionally include the requirements from FCS_TLSC_EXT.4. If the TOE implements session renegotiation, this selection must be made.

FCS_TLSC_EXT.1.2 The product shall verify that the presented identifier matches the reference identifier according to RFC 6125.

Application Note: The rules for verification of identity are described in Section 6 of RFC 6125. The reference identifier is established by the user (e.g. entering a URL into a web browser or clicking a link), by configuration (e.g. configuring the name of a mail server or authentication server), or by an application (e.g. a parameter of an API) depending on the product service. Based on a singular reference identifier's source domain and application service type (e.g. HTTP, SIP, LDAP), the client establishes all reference identifiers which are acceptable, such as a Common Name for the Subject Name field of the certificate and a (case-insensitive) DNS name, URI name, and Service Name for the Subject Alternative Name field. The client then compares this list of all acceptable reference identifiers to the presented identifiers in the TLS server's certificate.
The preferred method for verification is the Subject Alternative Name using DNS names, URI names, or Service Names. Verification using the Common Name for the purposes of backwards compatibility is optional. Additionally, support for use of IP addresses in the Subject Name or Subject Alternative name is discouraged, as against best practices, but may be implemented. Finally, the client should avoid constructing reference identifiers using wildcards. However, if the presented identifiers include wildcards, the client must follow the best practices regarding matching; these best practices are captured in the evaluation activity.

FCS_TLSC_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: Validity is determined by the identifier verification, certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1 as defined in any PP or PP-Module which instantiates this Package.

The selection that permits override for invalid certificates should be interpreted as follows:

- explicit administrator or user action is needed to authorize the override, on a per-certificate basis
- override may be sought or granted at any time, though this typically occurs when an invalid certificate is presented during connection setup
- override decisions may be stored and then consulted later, to permit connections using these otherwise-invalid certificates to establish trusted channels without user or administrator action

As indicated in , note that a PP author may instantiate this SFR using only the first selection, preventing the ability to allow overrides.

TLS Client Support for Mutual Authentication

FCS_TLSC_EXT.2.1 The product shall support mutual authentication using X.509v3 certificates.

Application Note: The use of X.509v3 certificates for TLS is addressed in FIA_X509_EXT.2.1. This requirement adds that a client must be capable of presenting a certificate to a TLS server for TLS mutual authentication. Presenting a certificate is not mandatory in all circumstances: it may depend on the configuration of the client or other factors.

TLS Client Support for Renegotiation

FCS_TLSC_EXT.4.1 The product shall support secure renegotiation through use of the "renegotiation_info" TLS extension in accordance with RFC 5746.

Application Note: RFC 5746 defines an extension to TLS that binds renegotiation handshakes to the cryptography in the original handshake.

Per RFC 5746, the client may present either the "renegotiation_info" extension or the signaling cipher suite value TLS_EMPTY_RENEGOTIATION_INFO_SCSV in the initial ClientHello message to indicate support for renegotiation. (A signaling cipher suite value (SCSV) is

presented as a cipher suite, but its only purpose is to provide other information and not to advertise support for a cipher suite.) The TLS_EMPTY_RENEGOTIATION_INFO_SCSV signaling cipher suite value exists as an alternative to presenting the "renegotation_info" extension so that TLS server implementations that immediately terminate the connection when they encounter any extension they do not understand can still proceed with a connection. The client may still choose to reject the connection later, if it insists upon renegotiation support and the server does not support it. In any case, RFC 5746 states that during any renegotiation the "renegotiation_info" extension must be presented by the peer initiating renegotiation, and so the client must support use of this extension.

TLS Client Support for Supported Groups Extension

FCS_TLSC_EXT.5.1 The product shall present the Supported Groups Extension in the Client Hello with the supported groups [**selection**:

- *secp256r1*,
- *secp384r1*,
- *secp521r1*,
- *ffdhe2048(256)*,
- *ffdhe3072(257)*,
- *ffdhe4096(258)*,
- *ffdhe6144(259)*,
- *ffdhe8192(260)*

].

Application Note: If an elliptic curve or Diffie-Hellman ciphersuite is selected in FCS_TLSC_EXT.1.1 or FCS_DTLSC_EXT.1.1, then FCS_TLSC_EXT.5 shall be included in the ST. This requirement does not limit the elliptic curves the client may propose for authentication and key agreement. The Supported Groups Extension was previously referred to as the Supported Elliptic Curves Extension and is described in RFC 7919.

TLS Server Protocol

FCS_TLSS_EXT.1.1 The product shall implement TLS 1.2 (RFC 5246) and [**selection**: *TLS 1.1 (RFC 4346)*, *no earlier TLS versions*] as a server that supports the cipher suites [**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 no other cipher suites, and also supports functionality for [**selection**:

- *mutual authentication*,
- *session renegotiation*,
- *none*

].

Application Note: The ST author should select the cipher suites that are supported, and must select at least one cipher suite. It is necessary to limit the cipher suites 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 cipher suites negotiated by the implementation are limited to those in this requirement, then the appropriate instructions need to be contained in the guidance. GCM cipher suites are preferred over CBC cipher suites, ECDHE preferred over RSA and DHE, and SHA256 or SHA384 over SHA.

TLS_RSA_WITH_AES_128_CBC_SHA is not required despite being mandated by RFC 5246.

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* is selected, then the ST must additionally include the requirements from FCS_TLSS_EXT.4. If the TOE implements session renegotiation, this selection must be made.

FCS_TLSS_EXT.1.2 The product shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0 and [**selection**: *TLS 1.1*, *none*].

Application Note: All SSL versions are denied. Any TLS version not selected in FCS_TLSS_EXT.1.1 should be selected here.

FCS_TLSS_EXT.1.3 The product shall perform key establishment for TLS 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 sizes]* ,
- *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 cipher suite in FCS_TLSS_EXT.1.1, the ST must include the RSA selection in the requirement.

If the ST lists a DHE cipher suite in FCS_TLSS_EXT.1.1, the ST must include either the Diffie-Hellman selection for parameters of a certain size, or for particular Diffie-Hellman groups. The selection for "Diffie-Hellman parameters" refers to the method defined by RFC 5246 (and RFC 4346) Section 7.4.3 where the server provides Diffie-Hellman parameters to the client. The Supported Groups extension defined in RFC 7919 identifies particular Diffie-Hellman groups, which are listed in the following selection. Regarding this distinction, it is acceptable to use Diffie-Hellman group 14 with TLS (there is currently no ability to negotiate group 14 using the Supported Groups extension, but it could be used with the "Diffie-Hellman parameters" selection). As in RFC 7919, the terms "DHE" and "FFDHE" are both used to refer to the finite-field-based Diffie-Hellman ephemeral key exchange mechanism, distinct from elliptic-curve-based Diffie Hellman ephemeral key exchange (ECDHE).

If the ST lists an ECDHE cipher suite in FCS_TLSS_EXT.1.1, the ST must include the selection for ECDHE using elliptic curves in the requirement.

TLS Server Support for Mutual Authentication

FCS_TLSS_EXT.2.1 The product shall support authentication of TLS clients using X.509v3 certificates.

FCS_TLSS_EXT.2.2 The product shall not establish a trusted channel if the client certificate is invalid.

Application Note: The use of X.509v3 certificates for TLS is addressed in FIA_X509_EXT.2.1 This requirement adds that this use must include support for client-side certificates for TLS mutual authentication. Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

FCS_TLSS_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: The client identifier may be in the Subject field or the Subject Alternative Name extension of the certificate. The expected identifier may either be configured, may be compared to the domain name, IP address, username, or email address used by the client, or may be passed to a directory server for comparison. In the latter case, the matching itself may be performed outside the TOE.

TLS Server Support for Renegotiation

FCS_TLSS_EXT.4.1 The product shall support the "renegotiation_info" TLS extension in accordance with RFC 5746.

FCS_TLSS_EXT.4.2 The product shall include the renegotiation_info extension in ServerHello messages.

Application Note: RFC 5746 defines an extension to TLS that binds renegotiation handshakes to the cryptography in the original handshake.

DTLS Client Protocol

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 cipher suites [**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 cipher suites 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 FCS_TLSC_EXT.1.2 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 FCS_TLSC_EXT.1.3 that are relevant to DTLS apply to this requirement.

FCS_DTLSC_EXT.1.4 The product shall [**selection**: *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.

DTLS Client Support for Mutual Authentication

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.

DTLS Server Protocol

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_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 no other cipher suites, 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 PP 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 PP, 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 cipher suite in FCS_DTLSS_EXT.1.1, the ST must include the RSA selection in the requirement.
If the ST lists a DHE cipher suite 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 cipher suite 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**: *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.

DTLS Server Support for Mutual Authentication

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 FCS_TLSS_EXT.2.1 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 FCS_TLSS_EXT.2.2 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 FCS_TLSS_EXT.2.3 that are relevant to DTLS apply to this requirement.

Objective Security Functional Requirements

TLS Client Support for Signature Algorithms Extension

FCS_TLSC_EXT.3.1 The product shall present the signature_algorithms extension in the Client Hello with the supported_signature_algorithms value containing the following hash algorithms: [**selection**: *SHA256*, *SHA384*, *SHA512*] and no other hash algorithms.

Application Note: This requirement limits the hashing algorithms supported for the purpose of digital signature verification by the client and limits the server to the supported hashes for the purpose of digital signature generation by the server. The signature_algorithms extension is only supported by TLS 1.2.

TLS Server Support for Signature Algorithms Extension

FCS_TLSS_EXT.3.1 The product shall present the HashAlgorithm enumeration in supported_signature_algorithms in the Certificate Request with the following hash algorithms: [**selection**: *SHA256*, *SHA384*, *SHA512*] and no other hash algorithms.

Application Note: This requirement limits the hashing algorithms supported for the purpose of digital signature verification by the server and limits the client to the supported hashes for the purpose of digital signature generation by the client. The supported_signature_algorithms is only supported by TLS 1.2.

Optional Security Functional Requirements