

## TLS Client Protocol (FCS\_TLSC)

**FCS\_TLSC\_EXT.1.1** The TSF shall implement TLS 1.2 (RFC 5246) and [selection: TLS 1.0(RFC 2246), TLS 1.1 (RFC 4346), no other] supporting the following ciphersuites: [

- Mandatory Ciphersuites:
  - TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 5246
- [selection: Optional Ciphersuites:
  - *TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 5246*
  - *TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 5246*
  - *TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 5246*
  - *TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492*
  - *TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 4492*
  - *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492*
  - *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 4492*
  - *TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5246*
  - *TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256 as defined in RFC 5246*
  - *TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5246*
  - *TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA256 as defined in RFC 5246*
  - *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5289*
  - *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384 as defined in RFC 5289*
  - *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5289*
  - *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5289*
  - *no other ciphersuite*]].

**Application Note:** The ciphersuites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional ciphersuites that are supported; if there are no ciphersuites supported other than the mandatory suites, then “None” should be selected. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. The Suite B algorithms listed above (RFC 6460) are the preferred algorithms for implementation. TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA is required in order to ensure compliance with RFC 5246.

TLS 1.0 and TLS 1.1 are currently allowed due to lack of support for TLS 1.2. TLS 1.0 and TLS 1.1 do not have the extensions necessary to assure a connection with security strength of 112-bits or better. The ST author should note in the TSS if the TOE provides additional hardening of TLS 1.0 and TLS 1.1 in order to meet this security strength.

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

If any ciphersuites are selected using ECDHE, then FCS\_TLS\_EXT.1.5 in Appendix A is required.

**Assurance Activity:**

The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component. By examining the TSS, the evaluator shall determine the additional hardening provided by the TOE in its implementation of the TLS protocol to assure security strengths. The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

The evaluator shall also perform the following tests:

- *Test 1:* The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an HTTPS 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 2:* The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.
- *Test 3:* The evaluator shall send a server certificate in the TLS connection that the does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA ciphersuite or send a RSA certificate while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.
- *Test 4:* The evaluator shall configure the server to select the TLS\_NULL\_WITH\_NULL\_NULL ciphersuite and verify that the client denies the connection.
- *Test 5:* The evaluator shall setup a man-in-the-middle tool between the TOE and the server or shall use customized server. The evaluator shall perform the following modifications to the traffic:
  - Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.
  - Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.

- Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.
- Modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.
- Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.
- Send an unencrypted packet from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.

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

**Application Note:** The rules for verification of identify are described in Section 6 of RFC 6125. The reference identifier is established by the user (e.g. entering a URL into a web browser or clicking a link), by configuration (e.g. configuring the name of a mail server or authentication server), or by an application (e.g. a parameter of an API) depending on the application service. Based on a singular reference identifier's source domain and application service type (e.g. HTTP, 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 is required for the purposes of backwards compatibility. Additionally, support for use of IP addresses in the Subject Name or Subject Alternative name is discouraged as against best practices but may be implemented. Finally, the client should avoid constructing reference identifiers using wildcards. However, if the presented identifiers include wildcards, the client must follow the best practices regarding matching; these best practices are captured in the assurance activity.

**Assurance Activity:**

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

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

The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection:

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

- *Test 7: [conditional]* If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

## TLS Server Protocol (FCS\_TLSS)

**FCS\_TLSS\_EXT.1.1** The TSF shall implement TLS 1.2 (RFC 5246) and [selection: TLS 1.0(RFC 2246), TLS 1.1 (RFC 4346), no other version] supporting the following ciphersuites: [

- Mandatory Ciphersuites:
  - TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 5246
- [selection: Optional Ciphersuites:
  - *TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 5246*
  - *TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 5246*
  - *TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 5246*
  - *TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492*
  - *TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 4492*
  - *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492*
  - *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 4492*
  - *TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5246*
  - *TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256 as defined in RFC 5246*
  - *TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5246*
  - *TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA256 as defined in RFC 5246*
  - *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5289*
  - *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384 as defined in RFC 5289*
  - *TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5289*
  - *TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5289*
  - *no other ciphersuite*]].

**Application Note (1):** The ciphersuites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional ciphersuites that are supported; if there are no ciphersuites supported other than the mandatory suites, then “None” should be selected. If administrative steps need to be taken so that the suites negotiated by the implementation are limited to those in this requirement, the appropriate instructions need to be contained in the guidance called for by AGD\_OPE. FMT\_SMF.1 addresses configuration of the ciphersuite to be used for connections.

**Application Note (2):** TLS 1.0 and TLS 1.1 are currently allowed due to lack of support for TLS 1.2. TLS 1.0 and TLS 1.1 do not have the extensions necessary to assure a connection with security strength of 112-bits or better. The ST author should note in the TSS if the TOE provides additional hardening of TLS 1.0 or TLS 1.1 in order to meet this security strength.

If ciphersuites with DHE or ECDHE are selected FCS\_TLSS\_EXT.1.4 in Appendix A shall be required.

**Assurance Activity:**

The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified are identical to those listed for this component. The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements), if required. By examining the TSS, the evaluator shall determine whether the TOE provides additional hardening in its implementation of the TLS protocol to assure security strengths.

The evaluator shall also perform the following tests:

- *Test 1:* The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS\_NULL\_WITH\_NULL\_NULL ciphersuite and verify that the server denies the connection.
- *Test 2:* The evaluator shall use a client to send a key exchange message in the TLS connection that does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after the receiving the key exchange message.
- *Test 3:* The evaluator shall setup a man-in-the-middle tool between the TOE and the server and shall perform the following modifications to the traffic:
  - Modify a byte in the client's nonce in the Client Hello handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.
  - Modify the signature block in the Client's Key Exchange handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.
  - Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.
  - After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection.
  - Send an unencrypted packet from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection.

**FCS\_TLSS\_EXT.1.2** The TSF shall deny connections from clients requesting SSL 1.0, SSL 2.0, SSL 3.0 and [selection: TLS 1.0, TLS 1.1, none].

**Application Note:** All SSL version shall be denied. Any TLS versions not selected in FCS\_TLSS\_EXT.1.1 should be selected here.

**Assurance Activity:**

The evaluator shall verify that the TSS contains a description of the denial of old SSL and selected TLS versions, and any configuration necessary to meet the requirement must be contained in the AGD guidance.

The evaluator shall also perform the following tests:

- *Test 4:* The evaluator shall send a Client Hello requesting a connection with version SSL 1.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 2.0, SSL 3.0, and any selected TLS versions.

**FCS\_TLSS\_EXT.1.3** The TSF shall not establish a trusted channel if the distinguished name (DN) or Subject Alternative Name (SAN) contained in a certificate does not match the expected identifier for the peer.

Application Note: This requirement only applies to those TOEs performing mutually-authenticated TLS. The peer identifier may be in the Subject field or the Subject Alternative Name extension of the certificate. The expected identifier may either be configured, may be compared to the Domain Name, IP address, username, or email address used by the peer, or may be passed to a directory server for comparison. Matching should be performed by a bit-wise comparison.

**Assurance Activity:**

*(Conditional)* If the TOE implements mutual authentication, the evaluator shall verify that the TSS describes how the DN and SAN in the certificate is compared to the expected identifier. If the DN is not compared automatically to the Domain Name, IP address, username, or email address, the evaluator shall ensure that the AGD guidance includes configuration of the expected identifier or the directory server for the connection.

The evaluator shall also perform the following tests:

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

**FMT\_SMF.1.1** The TSF shall be capable of performing the following security management functions:

- Configuring the security parameters to be used in TLS connections and
- [selection: *Enable/disable TLS renegotiation, no other*].



**Application Note:** The security parameters of the TLS connection include the ciphersuite and the algorithm strengths (hash algorithm, key agreement parameters) selected by the TLS Server. The ciphersuite for TLS connections may be configured by disabling all other ciphersuites or by creating a priority list. Implementations that use a priority list must still disable ciphersuites that are not allowed by this profile in the evaluated configuration, as indicated by FCS\_TLSS\_EXT.1.

**Assurance Activity:**

The evaluator shall review the administrator guidance to ensure that it contains instructions for configuring security parameters, including the algorithm strengths and ciphersuites, for TLS and shall perform the following test:

- If the guidance indicates that the configuration disables ciphersuites, the evaluator shall disable all but one ciphersuite, configure a client to send a Client Hello without that ciphersuite, and verify that the server denies an attempted connection.
- If the guidance indicates that the configuration establishes a priority list, the evaluator shall configure the priority list and the client such that the top priority ciphersuite is not in the Client Hello, and verify with a packet analyzer that the server picks the next priority ciphersuite during a connection. Additionally, the evaluator shall configure the client such that the top priority ciphersuite on the server is not the first ciphersuite presented in the Client Hello, and verify with a packet analyzer that the server picks the top priority ciphersuite during a connection.

## Appendix A

The below elements are required if certain selections are made within the SFR or if certain functionality exists in the TOE. The application notes include what selections or functionality result in these requirements.

### TLS Client Protocol (FCS\_TLSC)

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

**Application Note:** If non-HTTPS TLS connections are used, this component shall be claimed.

For TLS connections, this channel shall not be established if the peer certificate is invalid. The HTTPS protocol (FCS\_HTTPS\_EXT.1) requires different behavior, though HTTPS is implemented over TLS. This element addresses non-HTTPS TLS connections.

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.

**Assurance Activity:**

The evaluator shall use TLS as a function to verify that the validation rules in FIA\_X509\_EXT.1.1 are adhered to and shall perform the following additional test:

Test 1: The evaluator shall demonstrate that a peer using a certificate without a valid certification path results in an authenticate failure. Using the administrative guidance, the evaluator shall then load the trusted CA certificate(s) needed to validate the peer's certificate, and demonstrate that the connection succeeds. The evaluator then shall delete one of the CA certificates, and show that the connection fails.

**FCS\_TLSC\_EXT.1.4** The TSF shall support mutual authentication using X.509v3 certificates.

**Application Note:** If TLS is used for FTP\_ITC.1, then this component is required.

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

**Assurance Activity:**

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

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

The evaluator shall also perform the following test:

- *Test 1:* The evaluator shall setup a man-in-the-middle tool between the TOE and the server or shall use customized server. The evaluator shall perform the following modification to the traffic:
  - Configure the server to require mutual authentication and then modify a byte in a CA field in the Server's Certificate Request handshake message. The modified CA field must not be the CA used to sign the client's certificate. The evaluator shall verify the connection is unsuccessful.

**FCS\_TLSC\_EXT.1.5** The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [selection: *secp256r1*, *secp384r1*, *secp521r1*] and no other curves.

**Application Note:** If ciphersuites with elliptic curves were selected in FCS\_TLS\_EXT.1.1, this component is required.

This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS\_COP.1(3) and FCS\_CKM.1(1) and FCS\_CKM.1(2). This extension is required for clients supporting Elliptic Curve ciphersuites.

**Assurance Activity:**

The evaluator shall verify that TSS describes the Supported Elliptic Curves Extension and whether the required behavior is performed by default or may be configured. If the TSS indicates that the Supported Elliptic Curves Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves Extension.

The evaluator shall also perform the following tests:

- *Test 1:* The evaluator shall configure the server to such that the Server Hello contains an ECDHE ciphersuite. The evaluator shall then modify the server (or the server's packets) so that the key exchange message uses an ECDHE curve not supported by the client (per the supported curves extension) and shall verify that the TOE disconnects after receiving the server's key exchange handshake message.

## TLS Server Protocol (FCS\_TLSS)

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

**Application Note:** If the ST lists an DHE ciphersuite in FCS\_TLSS\_EXT.1.1, the ST must include the Diffie-Hellman selection in the requirement. If no DHE or ECDHE ciphersuite is selected, this component is not required.

FMT\_SMF.1 requires the configuration of the key agreement parameters in order to establish the security strength of the TLS connection.

### Assurance Activity:

The evaluator shall verify that the TSS describes the key agreement parameters of the server key exchange message, and any configuration necessary to meet the requirement must be contained in the AGD guidance.

The evaluator shall also perform the following tests:

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

**FCS\_TLSS\_EXT.1.5** The TSF shall support mutual authentication of TLS clients using X.509v3 certificates.

**Application Note:** If TLS is used for FPT\_ITC.1 then this component and FCS\_TLSS\_EXT.1.5 are required.

**FCS\_TLSS\_EXT.1.6** The TSF shall not establish a trusted channel if the peer 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.

**Assurance Activity:**

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

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

The evaluator shall perform the following mutual authentication tests:

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

Configure the server to require mutual authentication and then modify a byte in the client's Certificate Verify handshake message. The evaluator shall verify that the server rejects the connection.