first Daisy Diff compare report. last
Click on the changed parts for a detailed description. Use the left and right arrow keys to walk through the modifications.

Protection Profile for Application Software

This page is best viewed with JavaScript enabled!

Version: 1.~~2~~3

~~2016~~2019-~~04~~03-~~22~~01

National Information Assurance Partnership

Revision History

Version	Date	Comment
v 1.0	2014-10-20	Initial release
v 1.1	2014-11-05	Addition to TLS cipher suite selections Added server-side TLS requirements (selection-based)
v 1.2	2016-04-22	Multiple ~~clarifications~~ clarification based on NIAP TRRT inquiries Refactored FDP_DEC_EXT.1 into separate components
v 1.~~1~~3	~~2014~~2019-~~11~~03-~~05~~	~~Addition to TLS cipher suite selections~~ ~~v 1.0~~ ~~2014-10-20~~ ~~Initial release~~ 01 Incorporated available Technical Decisions Refactored FPT_TUD Added a selection to FTP_DIT Moved SWID Tags requirement Leveraged TLS Package Added equivalency section

Contents

1 Introduction

1.1 Overview

The scope of this Protection Profile (PP) is to describe the security functionality of application software in terms of ~~[CC]~~ and to define functional and assurance requirements for such software. In recent years, software attacks have shifted from targeting operating systems to targeting applications. This has been the natural response to improvements in operating system security and development processes. As a result, it is paramount that the security of applications be improved to reduce the risk of compromise.

1.2 Terms

The following sections list Common Criteria and technology terms used in this document. The following sections provide both Common Criteria and technology terms used in this Protection Profile.

1.2.1 Common Criteria Terms

Common Criteria (CC)	Common Criteria for Information Technology Security Evaluation.		
Common Evaluation Methodology (CEM)	Common Evaluation Methodology for Information Technology Security Evaluation.		
Extended Package (EP)	An implementation-independent set of security requirements for a specific subset of products described.		
Protection Profile (PP)	An implementation-independent set of security requirements for a category of products.		
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. In this case, application software and its supporting documentation.~~	~~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 a ST.		
~~Security Functional Requirement (SFR)~~	~~A requirement for security enforcement by the TOE.~~		
~~Security Assurance Requirement (SAR)~~	~~A requirement to assure the security of the TOE~~ Target of Evaluation (TOE)	The product under evaluation. In this case, the Operating System as described in section Section 1.3.1 TOE Boundary and its supporting documentation.	

1.2.2

~~Technology~~

Technical Terms

Address Space Layout Randomization (ASLR)	An anti-exploitation feature which loads memory mappings into unpredictable locations. ASLR makes it more difficult for an attacker to redirect control to code that they have introduced into the address space of an application process.
Application (app)	Software that runs on a platform and performs tasks on behalf of the user or owner of the platform, as well as its supporting documentation. The terms *TOE* and *application* are interchangeable in this document.
Application Programming Interface (API)	A specification of routines, data structures, object classes, and variables that allows an application to make use of services provided by another software component, such as a library. APIs are often provided for a set of libraries included with the platform.
Credential	Data that establishes the identity of a user, e.g. a cryptographic key or password.
Data Execution Prevention (DEP)	An anti-exploitation feature of modern operating systems executing on modern computer hardware, which enforces a non-execute permission on pages of memory. DEP prevents pages of memory from containing both data and instructions, which makes it more difficult for an attacker to introduce and execute code.
Developer	An entity that writes application software. For the purposes of this document, vendors and developers are the same.
Mobile Code	Software transmitted from a remote system for execution within a limited execution environment on the local system. Typically, there is no persistent installation and execution begins without the user's consent or even notification. Examples of mobile code technologies include JavaScript, Java applets, Adobe Flash, and Microsoft Silverlight.
Operating System (OS)	Software that manages hardware resources and provides services for applications.
Personally Identifiable Information (PII)	Any information about an individual maintained by an agency, including, but not limited to, education, financial transactions, medical history, and criminal or employment history and information which can be used to distinguish or trace an individual's identity, such as their name, social security number, date and place of birth, mother's maiden name, biometric records, etc., including any other personal information which is linked or linkable to an individual. [OMB]
Platform	The environment in which application software runs. The platform can be an operating system, ~~an~~ hardware environment, a software based execution environment ~~which runs atop an operating system~~, or some combination of these. These types platforms may also run atop other platforms.
Sensitive Data	Sensitive data may include all user or enterprise data or may be specific application data such as emails, messaging, documents, calendar items, and contacts. Sensitive data must minimally include PII, credentials, and keys. Sensitive data shall be identified in the application's TSS by the ST author.
Stack Cookie	An anti-exploitation feature that places a value on the stack at the start of a function call, and checks that the value is the same at the end of the function call. This is also referred to as Stack Guard, or Stack Canaries.
Vendor	An entity that sells application software. For purposes of this document, vendors and developers are the same. Vendors are responsible for maintaining and updating application software.

1.3 Compliant Targets of Evaluation

The requirements in this document apply to application software which runs on

~~mobile devices ("apps"), as well as on desktop and server platforms~~

any type of platform. Some application types are covered by more specific PPs, which may be expressed as

~~Extended Packages~~

PP-Modules of this PP. Such applications are subject to the requirements of both this PP and the

~~Extended Package~~

PP-Module that addresses their special functionality. PPs for some particularly specialized applications may not be expressed as

~~Extended Packages~~

PP-Modules at this time, though the requirements in this document should be seen as objectives for those highly specialized applications.

Although the requirements in this document apply to a wide range of application software, consult guidance from the relevant national schemes to determine when formal Common Criteria evaluation is expected for a particular type of application. This may vary depending upon the nature of the security functionality of the application.

1.3.1 TOE Boundary

The application, which consists of the software provided by its vendor, is installed onto the

~~filesystem provided by the operating system~~

platform(s) it operates on. It executes on the platform, which may be an operating system (Figure 1),

~~an~~

hardware environment, a software based execution environment, or some combination of these (Figure 2).

~~Some assurance~~

Those platforms may themselves run within other environments, such as virtual machines or operating systems, that completely abstract away the underlying hardware from the application. The TOE is not accountable for security functionality that is implemented by platform layers that are abstracted away. Some evaluation activities are specific to the particular platform on which the application runs, in order to provide precision and repeatability.

~~Test activities are actively sought from platform vendors so that coverage across platforms is as complete and accurate as possible. This will also enable certification of applications on those platforms.~~

The only platforms currently recognized by the AppPP are those specified in SFR Evaluation Activities. To test on a platform for which there are no EAs, a Vendor should contact NIAP with recommended EAs. NIAP will determine if the proposed platform is appropriate for the PP and accept, reject, or develop EAs as necessary in coordination with the technical community.

Applications include a diverse range of software such as office suites, thin clients, PDF readers,

~~and~~

downloadable smartphone apps, and apps running in a cloud container. The TOE includes any software in the application installation package, even those pieces that may extend or modify the functionality of the underlying platform, such as kernel drivers. Many platforms come bundled with applications such as web browsers, email clients and media players and these too should be considered subject to the requirements defined in this document although the expectation of formal Common Criteria evaluation depends upon the national scheme. BIOS and other firmware, the operating system kernel, and other systems software (and drivers) provided as part of the platform are outside the scope of this document.

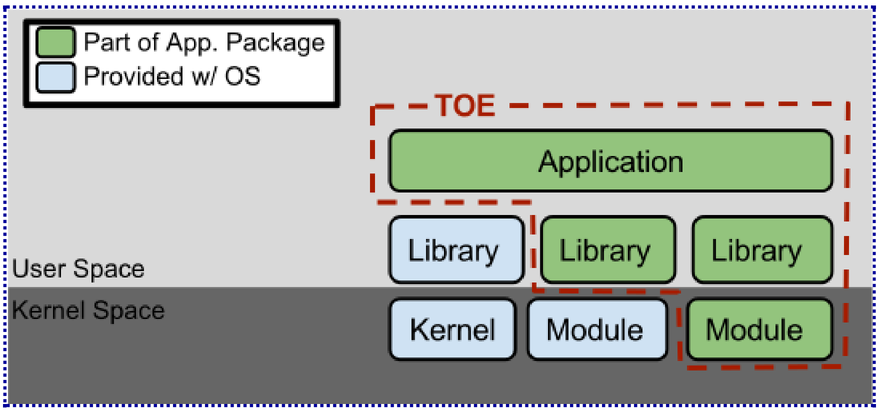

Figure 1: TOE as an Application and Kernel Module Running on an Operating System

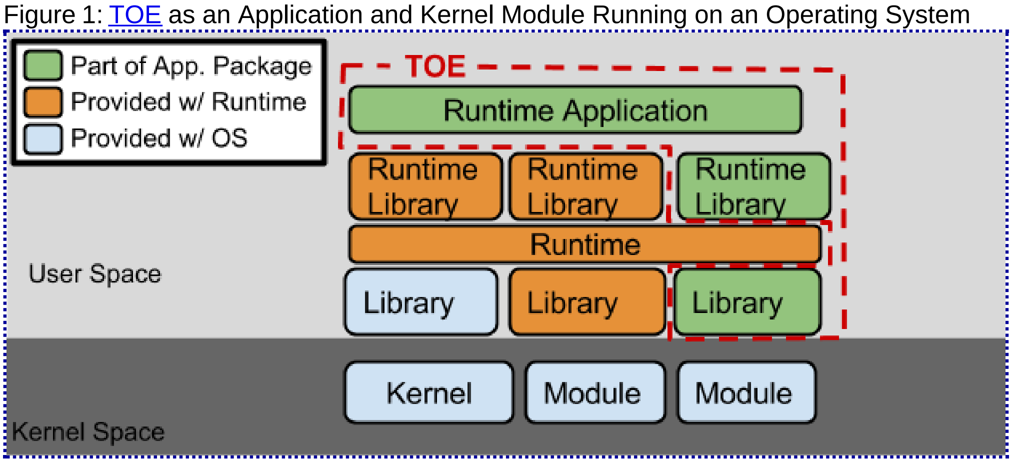

Figure 2: TOE as an Application Running in an Execution Environment Plus Native Code

1.4 Use Cases

Requirements in this Protection Profile are designed to address the security problem in the following use cases. These use cases are intentionally very broad, as many specific use cases exist for application software. Many applications may be used in combinations of these broad use cases, and evaluation against ~~Extended Packages~~ PP-Modules of this PP, when available, may be most appropriate for some application types.

[USE CASE 1] Content Creation
: The application allows a user to create content, saving it to either local or remote storage. Example content includes text documents, presentations, and images.

[USE CASE 2] Content Consumption
: The application allows a user to consume content, retrieving it from either local or remote storage. Example content includes web pages and video.

[USE CASE 3] Communication
: The application allows for communication interactively or non-interactively with other users or applications over a communications channel. Example communications include instant messages, email, and voice.

2

~~.~~

Conformance Claims

Conformance Statement

~~To be conformant~~ An ST must claim exact conformance to this PP, ~~a ST must demonstrate Exact Conformance, a subset of Strict Conformance~~ as defined in ~~[CC] Part 1 (ASE_CCL). The ST must include all components in this PP that are:~~

- ~~unconditional (which are always required)~~
- ~~selection-based (which are required when certain *selections* are chosen in the unconditional requirements)~~

~~and may include components that are~~

- ~~optional or~~
- ~~objective.~~

~~Unconditional requirements are found in the main body of the document, while appendices contain the selection-based, optional, and objective requirements. The ST may iterate any of these components, but it must not include any additional component (e.g. from CC Part 2 or 3 or a PP not conformant with this one, or extended by the ST) not defined in this PP or a PP conformant to this one. See Section 1.3 regarding more-specific PPs that may extend this one.~~ the CC and CEM addenda for Exact Conformance, Selection-Based SFRs, and Optional SFRs (dated May 2017).

CC Conformance Claims

This PP is conformant to Parts 2 (extended) and 3 (extended) of Common Criteria Version 3.1, Revision ~~4~~5.~~ [CC].~~

PP Claim

This PP does not claim conformance to any other Protection Profile.

The following PPs and PP-Modules are allowed to be specified in a PP-Configuration with this PP.

- PP-Module for File Encryption, Version 1.0
- PP-Module for File Encryption Enterprise Management, Version 1.0

Package Claim

This PP ~~does not claim conformance to any packages~~.

~~3. Security Problem Definition~~

is TLS Package Version 1.1 Conformant.

3 Security Problem Description

The security problem is described in terms of the threats that the TOE is expected to address, assumptions about the operational environment, and any organizational security policies that the TOE is expected to enforce.

3.1 Threats

T.NETWORK_ATTACK

An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may engage in communications with the application software or alter communications between the application software and other endpoints in order to compromise it.

T.NETWORK_EAVESDROP

An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may monitor and gain access to data exchanged between the application and other endpoints.

T.LOCAL_ATTACK

An attacker can act through unprivileged software on the same computing platform on which the application executes. Attackers may provide maliciously formatted input to the application in the form of files or other local communications.

T.PHYSICAL_ACCESS

An attacker may try to access sensitive data at rest.

3.2 Assumptions

A.PLATFORM

The TOE relies upon a trustworthy computing platform with a reliable time clock for its execution. This includes the underlying platform and whatever runtime environment it provides to the TOE.

A.PROPER_USER

The user of the application software is not willfully negligent or hostile, and uses the software in compliance with the applied enterprise security policy.

A.PROPER_ADMIN

The administrator of the application software is not careless, willfully negligent or hostile, and administers the software ~~within~~ in compliance ~~of~~ with the applied enterprise security policy.

3.3 Organizational Security Policies

There are no Organizational Security Policies for the application.

4

~~.~~

Security Objectives

4.1 Security Objectives for the TOE

O.INTEGRITY

Conformant ~~TOEs~~ TOEs ensure the integrity of their installation and update packages, and also leverage execution environment-based mitigations. Software is seldom, if ever, shipped without errors~~, and the~~. The ability to deploy patches and updates to fielded software with integrity is critical to enterprise network security. Processor manufacturers, compiler developers, execution environment vendors, and operating system vendors have developed execution environment-based mitigations that increase the cost to attackers by adding complexity to the task of compromising systems. Application software can often take advantage of these mechanisms by using ~~APIs~~ APIs provided by the runtime environment or by enabling the mechanism through compiler or linker options.

Addressed by: FDP_DEC_EXT.1, FMT_CFG_EXT.1, FPT_AEX_EXT.1~~,~~ FPT_TUD_EXT.1

O.QUALITY

To ensure quality of implementation, conformant ~~TOEs~~ TOEs leverage services and ~~APIs~~ APIs provided by the runtime environment rather than implementing their own versions of these services and APIs. This is especially important for cryptographic services and other complex operations such as file and media parsing. Leveraging this platform behavior relies upon using only documented and supported APIs.

Addressed by: FMT_MEC_EXT.1, FPT_API_EXT.1, FPT_API_EXT.2, FPT_LIB_EXT.1, FPT_TUD_EXT.2, FCS_CKM.1/1

O.MANAGEMENT

To facilitate management by users and the enterprise, conformant ~~TOEs~~ TOEs provide consistent and supported interfaces for their security-relevant configuration and maintenance. This includes the deployment of applications and application updates through the use of platform-supported deployment mechanisms and formats, as well as providing mechanisms for configuration. This also includes providing control to the user regarding disclosure of any PII.

Addressed by: FMT_SMF.1, FPT_IDV_EXT.1, FPT_TUD_EXT.1~~.5~~, FPR_ANO_EXT.1, FCS_COP.1/3

O.PROTECTED_STORAGE

To address the issue of loss of confidentiality of user data in the event of loss of physical control of the storage medium, conformant ~~TOEs~~ TOEs will use data-at-rest protection. This involves encrypting data and keys stored by the TOE in order to prevent unauthorized access to this data. This also includes unnecessary network communications whose consequence may be the loss of data.

Addressed by: FDP_DAR_EXT.1, ~~FDP~~FCS_~~DAR~~STO_EXT.1, FCS_~~STO~~RBG_EXT.1,FCS_~~RBG_EXT.1~~CKM.1/3, FCS_COP.1/1, FCS_COP.1/2, FCS_COP.1/4

O.PROTECTED_COMMS

To address both passive (eavesdropping) and active (packet modification) network attack threats, conformant~~TOEs~~ TOEs will use a trusted channel for sensitive data. Sensitive data includes cryptographic keys, passwords, and any other data specific to the application that should not be exposed outside of the application.

Addressed by: FTP_DIT_EXT.1, FCS_~~TLSC~~RBG_EXT.1, FCS_~~DTLS~~RBG_EXT.2, FCS_CKM_EXT.1, FCS_~~RBG_~~CKM.2, FCS_HTTPS_EXT.1, FDP_NET_EXT.1, FIA_X509_EXT.1

4.2 Security Objectives for the Operational Environment

The following security objectives for the operational environment assist the TOE in correctly providing its security functionality. These track with the assumptions about the environment.

OE.PLATFORM
: The TOE relies upon a trustworthy computing platform for its execution. This includes the underlying operating system and any discrete execution environment provided to the TOE.

OE.PROPER_USER
: The user of the application software is not willfully negligent or hostile, and uses the software within compliance of the applied enterprise security policy.

OE.PROPER_ADMIN
: The administrator of the application software is not careless, willfully negligent or hostile, and administers the software within compliance of the applied enterprise security policy.

4.3 Security Objectives Rationale

This section describes how the assumptions, threats, and ~~organizational~~ organization security policies map to the security objectives.

Threat, Assumption, or OSP	Security Objectives	Rationale
T.NETWORK_ATTACK	O.PROTECTED_COMMS, O.INTEGRITY, O.MANAGEMENT	The threat T.NETWORK_ATTACK is countered by O.PROTECTED_COMMS as this provides for integrity of transmitted data. The threat T.NETWORK_ATTACK is countered by O.INTEGRITY as this provides for integrity of software that is installed onto the system from the network. The threat T.NETWORK_ATTACK is countered by O.MANAGEMENT as this provides for the ability to configure the application to defend against network attack.
T.NETWORK_EAVESDROP	O.PROTECTED_COMMS, O.QUALITY, O.MANAGEMENT	The threat T.NETWORK_EAVESDROP is countered by O.PROTECTED_COMMS as this provides for confidentiality of transmitted data. The objective O.QUALITY ensures use of mechanisms that provide protection against network-based attack. The threat T.NETWORK_EAVESDROP is countered by O.MANAGEMENT as this provides for the ability to configure the application to protect the confidentiality of its transmitted data.
T.LOCAL_ATTACK	O.QUALITY	The objective O.QUALITY protects against the use of mechanisms that weaken the TOE with regard to attack by other software on the platform.
T.PHYSICAL_ACCESS	O.PROTECTED_STORAGE	The objective O.PROTECTED_STORAGE protects against unauthorized attempts to access physical storage used by the TOE.
A.PLATFORM	OE.PLATFORM	The operational environment objective OE.PLATFORM is realized through A.PLATFORM.
A.PROPER_USER	OE.PROPER_USER	The operational environment objective OE.PROPER_USER is realized through A.PROPER_USER.
A.PROPER_ADMIN	OE.PROPER_ADMIN	The operational environment objective OE.PROPER_ADMIN is realized through A.PROPER_ADMIN.

5

~~-~~

Security Requirements

This chapter describes the security requirements which have to be fulfilled by the ~~TOE~~product under evaluation. Those requirements comprise functional components from Part 2 and assurance components from Part 3 of [CC]. The following notations are used:

- **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: are identified with a number inside parentheses (e.g. "(1)")

5.1 Security Functional Requirements

The Security Functional Requirements included in this section are derived from Part 2 of the Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision ~~4~~5, with additional extended functional components.

5.1.1 Cryptographic Support (FCS)

FCS_RBG_EXT.1 Random Bit Generation Services

FCS_RBG_EXT.1.1
The application shall [**selection**:

- *use no DRBG functionality*,
- *invoke platform-provided DRBG functionality*,
- *implement DRBG functionality*

] for its cryptographic operations.
Application Note: The selection *invoke platform-provided DRBG functionality* should only be chosen for direct invocations of the platform DRBG, calls to platform protocols that may then call the platform's DRBG are not directly using DRBG functionality and should select *use no DRBG functionality*.

If *implement DRBG functionality* is chosen, then additional FCS_RBG_EXT.2 elements shall be included in the ST.

In this requirement, cryptographic operations include all cryptographic key generation/derivation/agreement, IVs (for certain modes), as well as protocol-specific random values.
~~Assurance Activity~~
Cryptographic operations in this requirement refer to the other cryptographic requirements in this PP, not additional functionality that is not in scope.
Evaluation Activity
TSS

*If **use no DRBG functionality** is selected, the evaluator shall inspect the application and its developer documentation and verify that the application needs no random bit generation services.*

*If **implement DRBG functionality** is selected, the evaluator shall ensure that additional FCS_RBG_EXT.2 elements are included in the ST.*

*If **invoke platform-provided DRBG functionality** is selected, the evaluator performs the following activities. The evaluator shall examine the TSS to confirm that it identifies all functions (as described by the SFRs included in the ST) that obtain random numbers from the platform RBG. The evaluator shall determine that for each of these functions, the TSS states which platform interface (API) is used to obtain the random numbers. The evaluator shall confirm that each of these interfaces corresponds to the acceptable interfaces listed for each platform below.*

It should be noted that there is no expectation that the evaluators attempt to confirm that the APIs are being used correctly for the functions identified in the TSS; the activity is to list the used APIs and then do an existence check via decompilation.

Tests
*If **invoke platform-provided DRBG functionality** is selected, the following tests shall be performed:*

The evaluator shall
~~then~~
decompile the application binary using
~~an~~
a decompiler suitable for the application (TOE). The evaluator shall search the output of the decompiler to determine that, for each API listed in the TSS, that API appears in the output. If the representation of the API does not correspond directly to the strings in the following list, the evaluator shall provide a mapping from the decompiled text to its corresponding API, with a description of why the API text does not directly correspond to the decompiled text and justification that the decompiled

text corresponds to the associated API.

~~It should be noted that there is no expectation that the evaluators attempt to confirm that the APIs are being used "correctly" for the functions identified in the TSS; the activity is to list the used APIs and then do an existence check via decompilation.~~

The following are the per-platform list of acceptable APIs:
For
~~BlackBerry: The evaluator shall verify that the application invokes Security Builder Crypto GSE. For~~
Android: *The evaluator shall verify that the application uses at least one of javax.crypto.KeyGenerator class or the java.security.SecureRandom class or /dev/random or /dev/urandom.*
For Windows: *The evaluator shall verify that rand_s, RtlGenRandom, BCryptGenRandom, or CryptGenRandom API is used for classic desktop applications. The evaluator shall verify*
~~that the System.Random API is used~~
the application uses the RNGCryptoServiceProvider class or derives a class from System.Security.Cryptography.RandomNumberGenerator API for Windows Universal Applications. It is only required that the API is called/invoked, there is no requirement that the API be used directly. In future versions of this document, CryptGenRandom may be removed as an option as it is no longer the preferred API per vendor documentation.
For iOS: *The evaluator shall verify that the application invokes SecRandomCopyBytes or uses /dev/random directly to acquire random.*
For Linux: *The evaluator shall verify that the application collects random from /dev/random or /dev/urandom.*
For Solaris: *The evaluator shall verify that the application collects random from /dev/random.*
For
~~Mac OS X~~
macOS: *The evaluator shall verify that the application*
~~uses~~
collects random from /dev/random
~~to acquire random~~
.
If invocation of platform-provided functionality is achieved in another way, the evaluator shall ensure the TSS describes how this is carried out, and how it is equivalent to the methods listed here (e.g. higher-level API invokes identical low-level API).

FCS_CKM_EXT.1 Cryptographic Key Generation Services

FCS_CKM_EXT.1.1
The application shall [**selection**:

- *generate no asymmetric cryptographic keys*,
- *invoke platform-provided functionality for asymmetric key generation*,
- *implement asymmetric key generation*

].
Application Note: If *implement asymmetric key generation* or *invoke platform-provided functionality for asymmetric key generation* is chosen, then additional FCS_CKM.1/1 elements shall be included in the ST.
Evaluation Activity
TSS
The evaluator shall inspect the application and its developer documentation to determine if the application needs asymmetric key generation services. If not, the evaluator shall verify the ***generate no asymmetric cryptographic keys*** *selection is present in the ST. Otherwise, the evaluation activities shall be performed as stated in the selection-based requirements.*

FCS_STO_EXT.1 Storage of Credentials

FCS_STO_EXT.1.1
The application shall [**selection**:

- *not store any credentials*,
- *invoke the functionality provided by the platform to securely store [**assignment**: list of credentials]* ,
- *implement functionality to securely store [**assignment**: list of credentials] according to [**selection**: FCS_COP.1(1), FCS_CKM.1(3)]*

] to non-volatile memory.
Application Note: This requirement ensures that persistent credentials (secret keys, PKI private keys,
~~or~~
passwords, etc) are stored securely
~~.~~

~~The assurance activity implicitly restricts which selections can be made, on per-platform basis. For example, if a platform provides~~
, and never persisted in cleartext form. Application developers are encouraged to use platform mechanisms for the secure storage of credentials. Depending on the platform that may include hardware-backed protection for credential storage. Application developers must choose a selection,

~~then the third selection cannot be indicated.~~

~~If~~
or multiple selections, based on all credentials that the application stores. If *not store any credentials* is selected then the application must not store any credentials. If *invoke the functionality provided by the platform to securely store* is selected then the Application developer must closely review the EA for their platform and provide documentation indicating which platform mechanisms are used to store credentials. If *implement functionality to securely store credentials* is selected, then the following components must be included in the ST: FCS_COP.1
~~(1).~~
/1 or FCS_CKM.1/3. If other cryptographic operations are used to implement the secure storage of credentials, the corresponding requirements must be included in the ST.
~~Assurance Activity~~ ➤
If the OS is Linux and Java KeyStores are used to store credentials, *implement functionality to securely store credentials* must be selected.
Evaluation Activity
TSS
The evaluator shall check the TSS to ensure that it lists all persistent credentials (secret keys, PKI private keys, or passwords) needed to meet the requirements in the ST. For each of these items, the evaluator shall confirm that the TSS lists for what purpose it is used, and how it is stored.
Tests
For all credentials for which the application implements functionality, the evaluator shall verify credentials are encrypted according to FCS_COP.1/1 or conditioned according to FCS_CKM.1.1/1 and FCS_CKM.1/3. For all credentials for which the application invokes platform-provided functionality, the evaluator shall perform the following actions which vary per platform.
For
~~BlackBerry: The evaluator shall verify that the application uses the BlackBerry KeyStore and Security Builder APIs to store credentials. For~~
Android: *The evaluator shall verify that the application uses the Android KeyStore or the Android KeyChain to store certificates.*
For Windows: *The evaluator shall verify that all certificates are stored in the Windows Certificate Store. The evaluator shall verify that other credentials, like passwords, are stored in the Windows Credential Manager or stored using the Data Protection API (DPAPI). For Windows Universal Applications, the evaluator shall verify that the application is using the ProtectData class and storing credentials in IsolatedStorage.*
For iOS: *The evaluator shall verify that all credentials are stored within a Keychain.*
For Linux: *The evaluator shall verify that all keys are stored using Linux keyrings.*
For Solaris: *The evaluator shall verify that all keys are stored using Solaris Key Management Framework (KMF).*
For
~~Mac OS X~~
macOS: *The evaluator shall verify that all credentials are stored within Keychain.*

5.1.2 User Data Protection (FDP)

FDP_DEC_EXT.1 Access to Platform Resources

FDP_DEC_EXT.1.1
The application shall restrict its access to [**selection**:

- *no hardware resources*,
- *network connectivity*,
- *camera*,
- *microphone*,
- *location services*,
- *NFC*,
- *USB*,
- *Bluetooth*,
- *[**assignment**: list of additional hardware resources]*

].
Application Note: The intent is for the evaluator to ensure that the selection captures all hardware resources which the application accesses, and that these are restricted to those which are justified. On some platforms, the application must explicitly solicit permission in order to access hardware resources. Seeking such permissions, even if the application does not later make use of the hardware resource, should still be considered access. Selections should be expressed in a manner consistent with how the application expresses its access needs to the underlying platform. For example, the platform may provide *location services* which implies the potential use of a variety of hardware resources (e.g. satellite receivers, WiFi, cellular radio) yet *location services* is the proper selection. This is because use of these resources can be inferred, but also because the actual usage may vary based on the particular platform. Resources that do not need to be explicitly identified are those which are ordinarily used by any application such as central processing units, main memory, displays, input devices (e.g. keyboards, mice), and persistent storage devices provided by the platform.
~~Assurance~~
Evaluation Activity
➤
Guidance
The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to hardware resources. The evaluator shall ensure that this is consistent with the selections indicated.

The evaluator shall review documentation provided by the application developer and for each resource which it accesses, identify the justification as to why access is required.
~~***For BlackBerry:*** The evaluator shall install the application and run it for the first time. The evaluator shall verify that the selection captures all hardware resources it would like to access. Note: If the user goes to: App permissions > Settings > Security and Privacy > Application Permissions > Select application in question, it will list which platform resource are approved/denied and can be changed.~~
Tests
For Android: *The evaluator shall inspect permissions presented at installation time (Android 5.1 and below) or on-access (Android 6.0 and above) for each hardware resource an [app](#) intends to access.*
For Windows: *For Windows Universal Applications the evaluator shall check the WMAppManifest.xml file for a list of required hardware capabilities. The evaluator shall verify that the user is made aware of the required hardware capabilities when the application is first installed. This includes permissions such as ID_CAP_ISV_CAMERA, ID_CAP_LOCATION, ID_CAP_NETWORKING, ID_CAP_MICROPHONE, ID_CAP_PROXIMITY and so on. A complete list of Windows App permissions can be found at:*

- *http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx*

For Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of the required
~~sensitive information repositories~~
hardware resources.
For iOS: *The evaluator shall verify that either the application or the documentation provides a list of the hardware resources it accesses.*
For Linux: *The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.*
For Solaris: *The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.*
For
~~Mac OS X~~
macOS: *The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.*
[FDP_DEC_EXT.1.2](#)
The application shall restrict its access to [**selection**:

- *no sensitive information repositories*,
- *address book*,
- *calendar*,
- *call lists*,
- *system logs*,
- *[**assignment**: list of additional sensitive information repositories]*

] .
Application Note: *Sensitive information repositories* are defined as those collections of sensitive data that could be expected to be shared among some applications, users, or user roles, but to which not all of these would ordinarily require access.
~~Assurance~~
[Evaluation Activity](#)
~~➤~~
Guidance
The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to sensitive information repositories. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each sensitive information
~~respository~~
repository which it accesses, identify the justification as to why access is required.
~~***For BlackBerry:*** The evaluator shall install the application and run it for the first time. The evaluator shall identify the sensitive information repositories to which it solicits access.~~
Tests
For Android: *The evaluator shall inspect permissions presented at installation time (Android 5.1 and below) or on-access (Android 6.0 and above) for each sensitive information repository an [app](#) intends to access.*
For Windows: *For Windows Universal Applications the evaluator shall check the WMAppManifest.xml file for a list of required capabilities. The evaluator shall identify the required information repositories when the application is first installed. This includes permissions such as ID_CAP_CONTACTS,ID_CAP_APPOINTMENTS,ID_CAP_MEDIALIB and so on. A complete list of Windows App permissions can be found at:*

- *http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx*

For Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of sensitive information repositories it accesses.
For iOS: *The evaluator shall verify that either the application software or its documentation provides a list of the sensitive information repositories it accesses.*
For Linux: *The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.*
For Solaris: *The evaluator shall verify that either the application software or its documentation provides a list of sensitive*

information repositories it accesses.
For
~~Mac OS X~~
macOS: *The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.*

FDP_NET_EXT.1 Network Communications

FDP_NET_EXT.1.1
The application shall restrict network communication to [**selection**:

- *no network communication*,
- *user-initiated communication for [**assignment**: list of functions for which the user can initiate network communication]*,
- *respond to [**assignment**: list of remotely initiated communication]*,
- *[**assignment**: list of application-initiated network communication]*

] .
Application Note: This requirement is intended to restrict both inbound and outbound network communications to only those required, or to network communications that are user initiated. It does not apply to network communications in which the application may generically access the filesystem which may result in the platform accessing remotely mounted drives/shares.
~~Assurance~~
Evaluation Activity
Tests
The evaluator shall perform the following tests:

- ***Test 1:*** *The evaluator shall run the application. While the application is running, the evaluator shall sniff network traffic ignoring all non-application associated traffic and verify that any network communications witnessed are documented in the TSS or are user-initiated.*
- ***Test 2:*** *The evaluator shall run the application. After the application initializes, the evaluator shall run network port scans to verify that any ports opened by the application have been captured in the ST for the third selection and its assignment. This includes connection-based protocols (e.g. TCP, DCCP) as well as connectionless protocols (e.g. UDP).*

For Android: *If "no network communication" is selected, the evaluator shall ensure that the application's AndroidManifest.xml file does not contain a*
~~<uses~~
uses-
~~permission>~~
permission or
~~<uses~~
uses-permission-sdk-
~~23>~~
23 tag containing android:name="android.permission.INTERNET". In this case, it is not necessary to perform the above Tests 1 and 2, as the platform will not allow the application to perform any network communication.

FDP_DAR_EXT.1 Encryption Of Sensitive Application Data

FDP_DAR_EXT.1.1
The application shall [**selection**:

- *leverage platform-provided functionality to encrypt sensitive data*,
- *implement functionality to encrypt sensitive data as defined in the PP-Module for File Encryption*,
- *protect sensitive data in accordance with FCS_STO_EXT.1*,
- *not store any sensitive data*

] in non-volatile memory.
Application Note: If *implement functionality to encrypt sensitive data as defined in the PP-Module for File Encryption* is selected,
~~then evaluation is required against the *Application Software Protection Profile Extended Package: File Encryption*~~
~~Assurance Activity~~
the TSF must claim conformance to a PP-Configuration that includes the File Encryption PP-Module.

Any file that may potentially contain sensitive data (to include temporary files) shall be protected. The only exception is if the user intentionally exports the sensitive data to non-protected files.
ST authors should select *protect sensitive data in accordance with FCS_STO_EXT.1* for the sensitive data that is covered by the FCS_STO_EXT.1 SFR.
Evaluation Activity
TSS
The evaluator shall
~~inventory the filesystem locations where the application may write data. The evaluator shall run the application and attempt~~

~~to store sensitive data~~
examine the [TSS](#) to ensure that it describes the sensitive data processed by the application. The evaluator shall then
~~inspect those areas of the filesystem to note where data was stored (if any), and determine whether it has been encrypte~~d
ensure that the following activities cover all of the sensitive data identified in the [TSS](#).

*If **not store any sensitive data** is selected, the evaluator shall inspect the [TSS](#)*
~~and~~
to ensure that it describes how sensitive data cannot be written to non-volatile memory. The evaluator shall also ensure that this is consistent with the filesystem test
~~above.~~

~~If **implement functionality to encrypt sensitive data** is selected, then evaluation is required against the ***Application Software Protection Profile Extended Package: File Encryption***. The evaluator shall ensure that such evaluation is underway.~~
below.

Tests

Evaluation activities (after the identification of the sensitive data) are to be performed on all sensitive data listed that are not covered by [FCS_STO_EXT.1](#).

The evaluator shall inventory the filesystem locations where the application may write data. The evaluator shall run the application and attempt to store sensitive data. The evaluator shall then inspect those areas of the filesystem to note where data was stored (if any), and determine whether it has been encrypted.

*If **leverage platform-provided functionality** is selected, the evaluation activities will be performed as stated in the following requirements, which vary on a per-platform basis*
~~: ***For BlackBerry:*** The evaluator shall inspect the TSS and ensure that it describes how the application uses the Advanced Data at Rest Protection [API](#) and how the application uses the appropriate domain to store and protect each data file~~
.

For Android: *The evaluator shall inspect the [TSS](#) and verify that it describes how files containing sensitive data are stored with the MODE_PRIVATE flag set.*
For Windows: *The Windows platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption, such as BitLocker or Encrypting File System (EFS), clear to the end user.*
For iOS: *The evaluator shall inspect the [TSS](#) and ensure that it describes how the application uses the Complete Protection, Protected Unless Open, or Protected Until First User Authentication Data Protection Class for each data file stored locally.*
For Linux: *The Linux platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.*
For Solaris: *The Solaris platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.*
For
~~Mac OS X~~
macOS: *The*
~~Mac OS X~~
macOS platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

5.1.3 Security Management (FMT)

FMT_MEC_EXT.1 Supported Configuration Mechanism

[FMT_MEC_EXT.1.1](#)
The application shall [**selection**: *invoke the mechanisms recommended by the platform vendor for storing and setting configuration options*, *implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the [PP](#)-Module for File Encryption*]
Application Note: Configuration options that are stored remotely are not subject to this requirement.
~~Assurance Activity~~
is generally not considered part of configuration options and should be stored according to [fdp_dar_ext.1](#) or [fcs_sto_ext.1](#).
[Evaluation Activity](#)
[TSS](#)
The evaluator shall review the [TSS](#) to identify the application's configuration options (e.g. settings) and determine whether these are stored and set using the mechanisms supported by the platform or implemented by the application in accordance with the [PP](#)-Module for File Encryption. At a minimum the [TSS](#) shall list settings related to any SFRs and any settings that are mandated in the operational guidance in response to an [SFR](#).
~~The method of doing so varies per platform. ***For BlackBerry:*** The evaluator shall run the application and make security-related changes to its configuration. The evaluator shall check that at least one file in the app folder of the application~~

~~working directory was modified to reflect the change made.~~

Conditional: If "implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption" is selected, the evaluator shall ensure that the TSS identifies those options, as well as indicates where the encrypted representation of these options is stored.
Tests
If "invoke the mechanisms recommended by the platform vendor for storing and setting configuration options" is chosen, the method of testing varies per platform as follows:
For Android: *The evaluator shall run the application and make security-related changes to its configuration. The evaluator shall check that at least one XML file at location /data/data/package/shared_prefs/ reflects the changes made to the configuration to verify that the application used* SharedPreferences *and/or* PreferenceActivity *classes for storing configuration data, where package is the Java package of the application.*
For Windows: *The evaluator shall determine and verify that Windows Universal Applications use either the Windows.UI.ApplicationSettings namespace or the IsolatedStorageSettings namespace for storing application specific settings. For .NET applications, the evaluator shall determine and verify that the application uses one of the locations listed in https://docs.microsoft.com/en-us/dotnet/framework/configure-apps/ for storing application specific settings. For Classic Desktop applications, the evaluator shall run the application while monitoring it with the*
~~SysInternal~~
SysInternals tool ProcMon and make changes to its configuration. The evaluator shall verify thatProcMon logs show corresponding changes to the the Windows Registry or C:\ProgramData\ directory.
For iOS: *The evaluator shall verify that the app uses the* user defaults system *or* key-value store *for storing all settings.*
For Linux: *The evaluator shall run the application while monitoring it with the utility strace. The evaluator shall make security-related changes to its configuration. The evaluator shall verify that strace logs corresponding changes to configuration files that reside in /etc (for system-specific configuration)*
~~or~~
, in the user's home directory (for user-specific configuration, or /var/lib/ (for configurations controlled by UI and not intended to be directly modified by an administrator).
For Solaris: *The evaluator shall run the application while monitoring it with the utility dtrace. The evaluator shall make security-related changes to its configuration. The evaluator shall verify that dtrace logs corresponding changes to configuration files that reside in /etc (for system-specific configuration) or in the user's home directory(for user-specific configuration).*
For
~~Mac OS X~~
macOS: *The evaluator shall verify that the application stores and retrieves settings using the* NSUserDefaults *class.*
If "implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption" is selected, for all configuration options listed in the TSS as being stored and protected using encryption, the evaluator shall examine the contents of the configuration option storage (identified in the TSS) to determine that the options have been encrypted.

FMT_CFG_EXT.1 Secure by Default Configuration

FMT_CFG_EXT.1.1
The application shall provide only enough functionality to set new credentials when configured with default credentials or no credentials.
Application Note: Default credentials are credentials (e.g., passwords, keys) that are automatically (without user interaction) loaded onto the platform during application installation. Credentials that are generated during installation using requirements laid out in FCS_RBG_EXT.1 are not by definition default credentials.
~~Assurance~~
Evaluation Activity
~~➤~~
TSS
The evaluator shall check the TSS to determine if the application requires any type of credentials and if the application installs with default credentials.
Tests
If the application uses any default credentials the evaluator shall run the following tests.

- ***Test 1:*** *The evaluator shall install and run the application without generating or loading new credentials and verify that only the minimal application functionality required to set new credentials is available.*
- ***Test 2:*** *The evaluator shall attempt to clear all credentials and verify that only the minimal application functionality required to set new credentials is available.*
- ***Test 3:*** *The evaluator shall run the application, establish new credentials and verify that the original default credentials no longer provide access to the application.*

FMT_CFG_EXT.1.2
The application shall be configured by default with file permissions which protect
~~it~~
the application binaries and
~~its~~
data
~~from unauthorized access~~
files from modification by normal unprivileged users

Application Note: The precise expectations for file permissions vary per platform but the general intention is that a trust boundary protects the application and its data.
~~Assurance~~
Evaluation Activity

Tests
The evaluator shall install and run the application. The evaluator shall inspect the filesystem of the platform (to the extent possible) for any files created by the application and ensure that their permissions are adequate to protect them. The method of doing so varies per platform.
For
~~BlackBerry~~
Android: *The evaluator shall run* *ls -alR|grep -E '^.......*
~~(r|-w|--x)~~
.w.' inside the application's data directories to ensure that all files are not world-
~~accessible (either read, write, or execute)~~
writable. The command should not print any files. The evaluator shall also verify that no sensitive data is written to external storage which could be read/modified by any
~~other~~
application
~~. ***For Android:*** The evaluator shall run ls -alR|grep -E '^.......(r|-w|--x)' inside the application's data directories to ensure that all files are not world-accessible (either read, write, or execute). The command should not print any files. The evaluator shall also verify that no sensitive data is written to external storage as this data can be read/modified by any application~~
containing the READ_EXTERNAL_STORAGE and/or WRITE_EXTERNAL_STORAGE permissions.
For Windows: *The evaluator shall run the SysInternals tools, Process Monitor and Access Check (or tools of equivalent capability, like icacls.exe) for Classic Desktop applications to verify that files written to disk during an*
~~applications~~
application's installation have the correct file permissions, such that a standard user cannot modify the application or its data files. For Windows Universal Applications the evaluator shall consider the requirement met because of the AppContainer sandbox.
For iOS: *The evaluator shall determine whether the application leverages the appropriate Data Protection Class for each data file stored locally.*
For Linux: *The evaluator shall run the command find . -perm /*
~~007~~
002 inside the application's data directories to ensure that all files are not world-
~~accessible (either read, write, or execute)~~
writable. The command should not print any files.
For Solaris: *The evaluator shall run the command find . \(-perm -*
~~001 -o -perm -~~
002
~~-o -perm -004~~
\) inside the application's data directories to ensure that all files are not world-
~~accessible (either read, write, or execute)~~
writable. The command should not print any files.
For
~~Mac OS X~~
macOS: *The evaluator shall run the command find . -perm +*
~~007~~
002 inside the application's data directories to ensure that all files are not world-
~~accessible (either read, write, or execute)~~
writable. The command should not print any files.

FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1
The TSF shall be capable of performing the following management functions [**selection**:

- *no management functions*,
- *enable/disable the transmission of any information describing the system's hardware, software, or configuration* ,
- *enable/disable the transmission of any PII* ,
- *enable/disable transmission of any application state (e.g. crashdump) information*,
- *enable/disable network backup functionality to [**assignment**: list of enterprise or commercial cloud backup systems]* ,
- *[**assignment**: list of other management functions to be provided by the TSF]*

] .
Application Note:
~~this~~
This requirement stipulates that an application needs to provide the ability to enable/disable only those functions that it actually implements.
~~the~~
The application is not responsible for controlling the behavior of the platform or other applications.
~~Assurance~~
Evaluation Activity

➤
Guidance
The evaluator shall verify that every management function mandated by the PP is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.
Tests
The evaluator shall test the application's ability to provide the management functions by configuring the application and testing each option selected from above. The evaluator is expected to test these functions in all the ways in which the ST and guidance documentation state the configuration can be managed.

5.1.4 Privacy (FPR)

FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information

FPR_ANO_EXT.1.1
The application shall [**selection**:

- *not transmit PII over a network ,*
- *require user approval before executing [**assignment**: list of functions that transmit PII over a network]*

] .
Application Note: This requirement applies only to PII that is specifically requested by the application; it does not apply if the user volunteers PII without prompting from the application into a general (or inappropriate) data field. A dialog box that declares intent to send PII presented to the user at the time the application is started is sufficient to meet this requirement.
~~Assurance~~
Evaluation Activity
➤
TSS
The evaluator shall inspect the TSS documentation to identify functionality in the application where PII can be transmitted
~~, and perform the following tests. **Test 1:** The~~
.
Tests
If require user approval before executing is selected, the evaluator shall run the application and exercise the functionality responsibly for transmitting PII and verify that user approval is required before transmission of the PII.

5.1.5 Protection of the TSF (FPT)

FPT_API_EXT.1 Use of Supported Services and APIs

FPT_API_EXT.1.1
The application shall use only documented platform
~~APIs~~
APIs.
Application Note: The definition of *documented* may vary depending upon whether the application is provided by a third party (who relies upon documented platform APIs) or by a platform vendor who may be able to guarantee support for platform APIs.
~~Assurance~~
Evaluation Activity
➤
TSS
The evaluator shall verify that the TSS lists the platform
~~APIs~~
APIs used in the application.
Tests
The evaluator shall then compare the list with the supported
~~APIs~~
APIs (available through e.g. developer accounts, platform developer groups) and ensure that all
~~APIs~~
APIs listed in the TSS are supported.

FPT_AEX_EXT.1 Anti-Exploitation Capabilities

FPT_AEX_EXT.1.1
The application shall not request to map memory at an explicit address except for [**assignment**: list of explicit exceptions].
Application Note: Requesting a memory mapping at an explicit address subverts address space layout randomization (ASLR).
~~Assurance~~
Evaluation Activity
➤
TSS
The evaluator shall ensure that the TSS describes the compiler flags used to enable ASLR when the application is compiled.

Tests
The evaluator shall perform either a static or dynamic analysis to determine that no memory mappings are placed at an explicit and consistent address. The method of doing so varies per platform.
For
~~BlackBerry~~
Android: *The evaluator shall run the same application on two different*
~~BlackBerry systems and run a tool that will list all memory mapped addresses for the application. The evaluator shall then verify the two different instances share no mapping locations.~~ ~~***For Android:*** The evaluator shall run the same application on two different Android systems.~~
Android systems. Both devices do not need to be evaluated, as the second device is acting only as a tool.Connect via [ADB]() and inspect /proc/[PID]()/maps. Ensure the two different instances share no
~~mapping locations.~~
memory mappings made by the application at the same location. Alternatively, the evaluator may use the same device. After collecting the first instance of mappings, the evaluator must uninstall the application, reboot the device, and reinstall the application to collect the second instance of mappings.
For Windows: *The evaluator shall run the same application on two different Windows systems and run a tool that will list all memory mapped addresses for the application. The evaluator shall then verify the two different instances share no mapping locations. The Microsoft*
~~sysinternals~~
SysInternals tool, VMMap, could be used to view memory addresses of a running application. The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application has [ASLR]() enabled.
For iOS: *The evaluator shall perform a static analysis to search for any mmap calls (or [API]() calls that call mmap), and ensure that no arguments are provided that request a mapping at a fixed address.*
For Linux: *The evaluator shall run the same application on two different Linux systems. The evaluator shall then compare their memory maps using* *pmap -x [PID]()* *to ensure the two different instances share no mapping locations.*
For Solaris: *The evaluator shall run the same application on two different Solaris systems. The evaluator shall then compare their memory maps using* *pmap -x [PID]()* *to ensure the two different instances share no mapping locations.*
For
~~Mac OS X~~
macOS: *The evaluator shall run the same application on two different Mac*
~~OS X~~
systems. The evaluator shall then compare their memory maps using *vmmap [PID]()* *to ensure the two different instances share no mapping locations.*
[FPT_AEX_EXT.1.2]()
The application shall [**selection**:

- *not allocate any memory region with both write and execute permissions*,
- *allocate memory regions with write and execute permissions for only [**assignment**: list of functions performing just-in-time compilation]*

] .
Application Note: Requesting a memory mapping with both write and execute permissions subverts the platform protection provided by [DEP](). If the application performs no just-in-time compiling, then the first selection must be chosen.
~~Assurance~~
[Evaluation Activity]()
➤
Tests
The evaluator shall verify that no memory mapping requests are made with write and execute permissions. The method of doing so varies per platform.
For
~~BlackBerry: The evaluator shall perform static analysis on the application to verify that~~

- ~~mmap is never invoked with both the PROT_WRITE and PROT_EXEC permissions, and~~
- ~~mprotect is never invoked.~~

~~For~~
Android: *The evaluator shall perform static analysis on the application to verify that*

- *mmap is never invoked with both the PROT_WRITE and PROT_EXEC permissions, and*
- *mprotect is never invoked.*

For Windows: *The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application passes the NXCheck. The evaluator may also ensure that the* */NXCOMPAT* *flag was used during compilation to verify that [DEP]() protections are enabled for the application.*
For iOS: *The evaluator shall perform static analysis on the application to verify that mprotect is never invoked with the PROT_EXEC permission.*
For Linux: *The evaluator shall perform static analysis on the application to verify that both*

- *mmap is never be invoked with both the PROT_WRITE and PROT_EXEC permissions, and*
- *mprotect is never invoked with the PROT_EXEC permission.*

For Solaris: *The evaluator shall perform static analysis on the application to verify that both*

- *mmap is never be invoked with both the PROT_WRITE and PROT_EXEC permissions, and*
- *mprotect is never invoked with the PROT_EXEC permission.*

For
~~Mac OS X~~
macOS: *The evaluator shall perform static analysis on the application to verify that mprotect is never invoked with the PROT_EXEC permission.*
FPT_AEX_EXT.1.3
The application shall be compatible with security features provided by the platform vendor.
Application Note: This requirement is designed to ensure that platform security features do not need to be disabled in order for the application to run.
~~Assurance~~
Evaluation Activity
Tests
The evaluator shall configure the platform in the ascribed manner and carry out one of the prescribed tests:
For
~~BlackBerry: The evaluator shall ensure that the application can successfully run on the latest version of the BlackBerry OS.~~
~~***For Android:*** The~~
Android: *Applications running on Android cannot disable Android security features, therefore this requirement is met and no evaluation activity is required.*
For Windows: *If the OS platform supports Windows Defender Exploit Guard (Windows 10 version 1709 or later), then the evaluator shall ensure that the application can*
~~successfully run on the latest version of Android.***For Windows:*** For both classic desktop and Windows Universal Applications, the evaluator shall configure the latest version of Microsoft's~~
run successfully with Windows Defender Exploit Guard Exploit Protection configured with the following minimum mitigations enabled; Control Flow Guard (CFG), Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), Import address filtering (IAF), and Data Execution Prevention (DEP). The following link describes how to enable Exploit Protection, https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-exploit-guard/customize-exploit-protection.

If the OS platform supports the Enhanced Mitigation Experience Toolkit (EMET)
~~to protect the application. The evaluator shall then run the application and verify that the application does not crash while protected by EMET. ***For iOS:*** The evaluator shall ensure that the application can successfully run on the latest version of iOS~~
which can be installed on Windows 10 version 1703 and earlier, then the evaluator shall ensure that the application can run successfully with EMET configured with the following minimum mitigations enabled; Memory Protection Check, Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), and Data Execution Prevention DEP).
For iOS: *Applications running on iOS cannot disable security features, therefore this requirement is met and no evaluation activity is required.*
For Linux: *The evaluator shall ensure that the application can successfully run on a system with either SELinux or AppArmor enabled and*
~~enforcing~~
in enforce mode.
For Solaris: *The evaluator shall ensure that the application can run with Solaris Trusted Extensions enabled and enforcing.*
For
~~Mac OS X~~
macOS: *The evaluator shall ensure that the application can successfully run on*
~~the latest version of OS X~~
macOS without disabling any security features.
FPT_AEX_EXT.1.4
The application shall not write user-modifiable files to directories that contain executable files unless explicitly directed by the user to do so.
Application Note: The purpose of this requirement is to help ensure the integrity of application binaries by supporting file protection mechanisms such as directory-level file permissions and application whitelisting. A user-modifiable file for purposes of this requirement is a file that is writable by an unprivileged user of the application -- either directly through application execution or independently of the application. If the application runs in the context of the application user, then the application should not be able to write to the directory containing the application binaries -- regardless of whether the files are configuration data, audit data, or temporary files. Executables and user-modifiable files may not share the same parent directory, but may share directories above the parent.
~~Assurance~~
Evaluation Activity
Tests
The evaluator shall run the application and determine where it writes its files. For files where the user does not choose the destination, the evaluator shall check whether the destination directory contains executable files. This varies per platform:
For
~~BlackBerry: The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox). For~~
Android: *The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored under /data/data/package/ where package is the Java package of the application.*

For Windows: *For Windows Universal Applications the evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox). For Windows Desktop Applications the evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote*
~~and no data files in the application's install directory~~
user-modifiable files.
For iOS: *The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).*
For Linux: *The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.*
For Solaris: *The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.*
For
~~Mac OS X~~
macOS: *The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.*
FPT_AEX_EXT.1.5
The application shall be
~~compiled~~
built with stack-based buffer overflow protection enabled.
~~Assurance~~
Evaluation Activity
➤
~~The evaluator shall ensure that the TSS section of the ST describes the compiler flag used to enable stack-based buffer overflow protection in the application. The evaluator shall perform a static analysis to verify~~
Tests
The evaluator will inspect every native executable included in the TOE to ensure that stack-based buffer overflow protection is present.
~~The method of doing so varies per platform:~~
For
~~BlackBerry: The evaluator shall ensure that the -fstack-protector-strong or -fstack-protector-all flags are used. The -fstack-protector-all flag is preferred but -fstack-protector-strong is acceptable.~~
~~***For Android:*** Applications that are entirely Java run in the Java machine and do not need traditional stack protection. For applications using Java Native Interface (JNI), the evaluator shall ensure that the -fstack-protector-strong or -fstack-protector-all flags are used. The -fstack-protector-all flag is preferred but -fstack-protector-strong is acceptable.~~
~~***For Windows:*** The~~
Windows: *Applications that run as Managed Code in the .NET Framework do not require these stack protections. Applications developed in Object Pascal using the Delphi IDE compiled with RangeChecking enabled comply with this element. For other code, the evaluator shall review the TSS and verify that the /GS flag was used during compilation. The evaluator shall run a tool like,*
~~like~~
BinScope, that can verify the correct usage of /GS.
For
~~iOS: If the application is compiled using GCC or Xcode~~
PE, *the evaluator*
~~shall~~
will disassemble each and ensure
~~that the -fstack-protector-strong or -fstack-protector-all flags are used. The -fstack-protector-all flag is preferred but -fstack-protector-strong is acceptable. If the application is built using any other compiler, then the evaluator shall determine that appropriate stack-protection has been used during the build process.~~
~~***For Linux:*** If the application is compiled using GCC, the evaluator shall ensure that the -fstack-protector-strong or -fstack-protector-all flags are used. The -fstack-protector-all flag is preferred but -fstack-protector-strong is acceptable. If the application is built using clang, it must be compiled and linked with the -fsanitize=address flag. If the application is built using any other compiler, then the evaluator shall determine that appropriate stack-protection has been used during the build process.~~
~~***For Solaris:*** If the application is compiled using GCC, the evaluator shall ensure that the -fstack-protector-strong or -fstack-protector-all flags are used. The -fstack-protector-all flag is preferred but -fstack-protector-strong is acceptable. If the application is built using clang, it must be compiled and linked with the -fsanitize=address flag. If the application is built using any other compiler, then the evaluator shall determine that appropriate stack-protection has been used during the build process.~~
~~***For Mac OS X:*** If the application is compiled using GCC or Xcode, the evaluator shall ensure that the -fstack-protector-strong or -fstack-protector-all flags are used. The -fstack-protector-all flag is preferred but -fstack-protector-strong is acceptable. If the application is built using any other compiler, then the evaluator shall determine that appropriate stack-protection has been used during the build process.~~
the following sequence appears:

```
mov rcx, QWORD PTR
[rsp+(...)]
xor rcx, (...)
call (...)
```

.
***For [ELF](#) executables**, the evaluator will ensure that each contains references to the symbol **__stack_chk_fail**.*

Tools such as [Canary Detector](#) may help automate these activities.

FPT_TUD_EXT.1 Integrity for Installation and Update

FPT_TUD_EXT.1.1
The application shall [**selection**: *provide the ability*, *leverage the platform*] to check for updates and patches to the application software.
Application Note: This requirement is about the ability to "check" for updates. The actual installation of any updates should be done by the platform. This requirement is intended to ensure that the application can check for updates provided by the vendor, as updates provided by another source may contain malicious code.
~~Assurance~~
Evaluation Activity
Guidance
The evaluator shall check to ensure the guidance includes a description of how updates are performed.
Tests
The evaluator shall check for an update using procedures described in either the application documentation or the platform documentation and verify that the application does not issue an error. If it is updated or if it reports that no update is available this requirement is considered to be met.
FPT_TUD_EXT.1.2
The application shall
~~be distributed using the format of the platform-supported package manager.~~
~~Assurance Activity~~
~~*The evaluator shall verify that application updates are distributed in the format supported by the platform. This varies per platform:*~~
~~***For BlackBerry:** The evaluator shall ensure that the application is packaged in the Blackberry (BAR) format.*~~
~~***For Android:** The evaluator shall ensure that the application is packaged in the Android application package (APK) format.*~~
~~***For Windows:** The evaluator shall ensure that the application is packaged in the standard Windows Installer (.MSI) format, the Windows Application Software (.EXE) format signed using the Microsoft Authenticode process, or the Windows Universal Application package (.APPX) format. See https://msdn.microsoft.com/en-us/library/ms537364(v=vs.85).aspx for details regarding Authenticode signing.*~~
~~***For iOS:** The evaluator shall ensure that the application is packaged in the IPA format.*~~
~~***For Linux:** The evaluator shall ensure that the application is packaged in the format of the package management infrastructure of the chosen distribution. For example, applications running on Red Hat and Red Hat derivatives should be packaged in RPM format. Applications running on Debian and Debian derivatives should be packaged in deb format.*~~
~~***For Solaris:** The evaluator shall ensure that the application is packaged in the PKG format.*~~
~~***For Mac OS X:** The evaluator shall ensure that application is packaged in the DMG format, the PKG format, or the MPKG format.*~~
~~FPT_TUD_EXT.1.3~~
~~The application shall be packaged such that its removal results in the deletion of all traces of the application, with the exception of configuration settings, output files, and audit/log events.~~
~~**Application Note:** Applications bundled with the system/firmware image are not subject to this requirement if the user is unable to remove the application through means provided by the OS.~~
~~Assurance Activity~~
~~*The evaluator shall record the path of every file on the entire filesystem prior to installation of the application, and then install and run the application. Afterwards, the evaluator shall then uninstall the application, and compare the resulting filesystem to the initial record to verify that no files, other than configuration, output, and audit/log files, have been added to the filesystem.*~~
[**selection**: *provide the ability*, *leverage the platform*] to query the current version of the application software.
Evaluation Activity
Guidance
The evaluator shall verify guidance includes a description of how to query the current version of the application.
Tests
The evaluator shall query the application for the current version of the software according to the operational user guidance. The evaluator shall then verify that the current version matches that of the documented and installed version.
FPT_TUD_EXT.1.
~~4~~
3
The application shall not download, modify, replace or update its own binary code.
Application Note: This requirement applies to the code of the application; it does not apply to mobile code technologies that are designed for download and execution by the application.
~~Assurance~~
Evaluation Activity
Tests
The evaluator shall verify that the application's executable files are not changed by the application. The evaluator shall complete the following test:

- ***Test 1:*** *The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the*

~~TSS~~

- *ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.*

FPT_TUD_EXT.1.
~~5~~
4
The application
~~shall [**selection**, at least one of: *provide the ability*, *leverage the platform*] to query the current version of the application software.~~
~~Assurance Activity~~
~~*The evaluator shall query the application for the current version of the software according to the operational user guidance (AGD_OPE.1) and shall verify that the current version matches that of the documented and installed version.*~~
~~FPT_TUD_EXT.1.6~~
~~The application~~
installation package and its updates shall be digitally signed such that its platform can cryptographically verify them prior to installation.
Application Note: The specifics of the verification of installation packages and updates involves requirements on the platform (and not the application), so these are not fully specified here.
~~Assurance~~
Evaluation Activity

TSS
The evaluator shall verify that the TSS identifies how the application installation package and updates to it are signed by an authorized source. The definition of an authorized source must be contained in the TSS. The evaluator shall also ensure that the TSS (or the operational guidance) describes how candidate updates are obtained.
FPT_TUD_EXT.1.5
The application is distributed [**selection**: *with the platform OS* , *as an additional software package to the platformOS*].
Application Note: Application software that is distributed as part of the platform operating system is not required to be package for installation or uninstallation. If "as an additional software package to the OS" is selected the requirements from FPT_TUD_EXT.2 must be included in the ST.
Evaluation Activity
TSS
The evaluator shall verify that the TSS identifies how the application is distributed. If "with the platform" is selected the evaluated shall perform a clean installation or factory reset to confirm that TOE software is included as part of the platform OS. If "as an additional package" is selected the evaluator shall perform the tests inFPT_TUD_EXT.2.

FPT_LIB_EXT.1 Use of Third Party Libraries

FPT_LIB_EXT.1.1
The application shall be packaged with only [**assignment**: list of third-party libraries].
Application Note: The intention of this requirement is for the evaluator to discover and document whether the application is including unnecessary or unexpected third-party libraries. This includes adware libraries which could present a privacy threat, as well as ensuring documentation of such libraries in case vulnerabilities are later discovered.
~~Assurance~~
Evaluation Activity

Tests
The evaluator shall install the application and survey its installation directory for dynamic libraries. The evaluator shall verify that libraries found to be packaged with or employed by the application are limited to those in the assignment.

FPT_IDV_EXT.1 Software Identification and Versions

FPT_IDV_EXT.1.1
The application shall be versioned with [**selection**: *SWID tags that comply with minimum requirements fromISO/IEC 19770-2:2015* , [***assignment****: other version information*]] .
Application Note: The use of SWID tag to identify application software is a requirement for DODIT based on DoD Instruction 8500.01 which requires the use of SCAP which includes SWID tags per the NIST standard. The PP selection of "other version information" will be removed in the next major release of this protection profile. Vendors should begin to version software with valid SWID tags.

Valid SWID tags must contain a SoftwareIdentity element and an Entity element as defined in theISO/IEC 19770-2:2015 standard. SWID tags must be stored with a .swidtag file extensions as defined in theISO/IEC 19770-2:2015.
Evaluation Activity
TSS
If "other version information" is selected the evaluator shall verify that the TSS contains an explaination of the versioning

methodology.
Tests
The evaluator shall install the application, then check for the / existence of version information. If SWID tags is selected the evaluator shall check for a .swidtag file. The evaluator shall open the file and verify that is contains at least a SoftwareIdentity element and an Entity element.

5.1.6 Trusted Path/Channel (FTP)

FTP_DIT_EXT.1 Protection of Data in Transit

FTP_DIT_EXT.1.1
The application shall [**selection**:

- *not transmit any [**selection**: data,*

~~not transmit any~~

- *sensitive data]* ,
- *encrypt all transmitted [**selection**: sensitive data, data] with [**selection***

~~, at least one of~~

- *: HTTPS in accordance with FCS_HTTPS_EXT.1, TLS as defined in the TLS Package, DTLS as defined in the TLS Package, SSH as conforming to the Extended Package for Secure Shell, IPsec as defined in the PP-Module for VPN Client]* ,
- *invoke platform-provided functionality to encrypt all transmitted sensitive data with [**selection***

~~, at least one of~~

- *: HTTPS, TLS, DTLS, SSH]* ,
- *invoke platform-provided functionality to encrypt all transmitted data with **selection**: HTTPS, TLS, DTLS, SSH]*

] between itself and another trusted IT product.
Application Note:
~~Extended packages may override this requirement to provide for other protocols.~~
Encryption is not required for applications transmitting data that is not sensitive.

If *encrypt all transmitted* is selected and *TLS* is selected, then evaluation of elements from either FCS_TLSC_EXT.1 or FCS_TLSS_EXT.1 is required.

If *encrypt all transmitted* is selected and *HTTPS* is selected,

~~then evaluation of elements from~~

FCS_HTTPS_EXT.1 is required.

If *encrypt all transmitted* is selected and *DTLS* is selected,

~~then evaluation of elements from~~

FCS_DTLS_EXT.1 is required.

If *encrypt all transmitted* is selected and *SSH* is selected, the TSF shall be validated against the *Extended Package for Secure Shell*.

~~Assurance Activity~~

If *encrypt all trasnmitted* is selected and *IPsec* is selected, the TSF must claim conformance to a *PP-Configuration that includes the VPN Client PP-Module*

If *encrypt all transmitted* is selected the corresponding FCS_COP.1 requirements will be included.

Evaluation Activity
TSS
For platform-provided functionality, the evaluator shall verify the TSS contains the calls to the platform that TOE is leveraging to invoke the functionality.
Tests
The evaluator shall perform the following tests.

- ***Test 1:** The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall verify from the packet capture that the traffic is encrypted with HTTPS, TLS*

~~or DTLS~~

- *, DTLS, or SSH, or IPsec in accordance with the selection in the ST.*
- ***Test 2:** The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall review the packet capture and verify that no sensitive data is transmitted in the clear.*
- ***Test 3:** The evaluator shall inspect the TSS to determine if user credentials are transmitted. If credentials are transmitted the evaluator shall set the credential to a known value. The evaluator shall capture packets from the application while causing credentials to be transmitted as described in the TSS. The evaluator shall perform a string search of the captured network packets and verify that the plaintext credential previously set by the evaluator is not found.*

***For Android:** If "not transmit any data" is selected, the evaluator shall ensure that the application's AndroidManifest.xml file does not contain a*
~~<uses~~
uses-
~~permission>~~
permission or
~~<uses~~
uses-permission-sdk-
~~23>~~
23 tag containing android:name="android.permission.INTERNET". In this case, it is not necessary to perform the above Tests 1, 2, or 3, as the platform will not allow the application to perform any network communication.
***For iOS:** If "encrypt all transmitted data" is selected, the evaluator shall ensure that the application's Info.plist file does not contain the NSAllowsArbitraryLoads or NSExceptionAllowsInsecureHTTPLoads keys, as these keys disable iOS's Application Transport Security feature.*

5.2 Security Assurance Requirements

The Security Objectives for the TOE in Section 5 Security Requirements were constructed to address threats identified in Section 3.1 Threats. The Security Functional Requirements (~~SFRs~~SFRs) in Section 5.1 Security Functional Requirements are a formal instantiation of the Security Objectives. The PP identifies the Security Assurance Requirements (~~SARs~~SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

This section lists the set of ~~SARs~~ SARs from CC part 3 that are required in evaluations against this PP. Individual ~~Assurance~~ Evaluation Activities (~~AAs~~EAs) to be performed are specified both in Section 5 Security Requirements as well as in this section.

The general model for evaluation of ~~TOEs~~ TOEs against STs written to conform to this PP is as follows:
After the ST has been approved for evaluation, the ~~Information Technology Security Evaluation Facility (ITSEF)~~ CCTL will obtain the TOE, supporting environmental IT, and the administrative/user guides for the TOE. The ~~ITSEF~~ CCTL is expected to perform actions mandated by the Common Evaluation Methodology (CEM) for the ASE and ALC SARs. The ~~ITSEF~~ CCTL also performs the ~~Assurance Activities~~ evaluation activities contained within Section 5 Security Requirements, which are intended to be an interpretation of the other CEM assurance requirements as they apply to the specific technology instantiated in the TOE. The ~~Assurance Activities~~ evaluation activities that are captured in Section 5 Security Requirements also provide clarification as to what the developer needs to provide to demonstrate the TOE is compliant with the PP. The results of these activities will be documented and presented (along with the administrative guidance used) for validation.

5.2.1 Class ASE: Security Target

As per ASE activities defined in [CEM].

5.2.2 Class ADV: Development

The information about the TOE is contained in the guidance documentation available to the end user as well as the TSS portion of the ST. The TOE developer must concur with the description of the product that is contained in the TSS as it relates to the functional requirements. The ~~Assurance Activities~~ evaluation activities contained in Section 5.1 Security Functional Requirements should provide the ST authors with sufficient information to determine the appropriate content for the TSS section.

ADV_FSP.1 Basic Functional Specification (ADV_FSP.1)

Developer action elements:

ADV_FSP.1.1D
The developer shall provide a functional specification.
ADV_FSP.1.2D
The developer shall provide a tracing from the functional specification to the SFRs.
Application Note: As indicated in the introduction to this section, the functional specification is comprised of the information contained in the AGD_OPE and AGD_PRE documentation. The developer may reference a website accessible to

application developers and the evaluator. The
~~assurance~~
evaluation activities in the functional requirements point to evidence that should exist in the documentation and TSS section; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.

Content and presentation elements:

ADV_FSP.1.
~~1C~~
3C
The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.
ADV_FSP.1.
~~2C~~
4C
The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.
ADV_FSP.1.
~~3C~~
5C
The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.
ADV_FSP.1.
~~4C~~
6C
The tracing shall demonstrate that the SFRs trace to
~~TSFIs~~
TSFIs in the functional specification.

Evaluator action elements:

ADV_FSP.1.
~~1E~~
7E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_FSP.1.
~~2E~~
8E
The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.
~~Assurance~~
Evaluation Activity

TSS
There are no specific
~~assurance~~
evaluation activities associated with these SARs, except ensuring the information is provided. The functional specification documentation is provided to support the evaluation activities described in Section 5.1 Security Functional Requirements, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other
~~assurance~~
evaluation activities being performed; if the evaluator is unable to perform an activity because there is insufficient interface information, then an adequate functional specification has not been provided.

5.2.3 Class AGD: Guidance Documentation

The guidance documents will be provided with the ST. Guidance must include a description of how the IT personnel verifies that the Operational Environment can fulfill its role for the security functionality. The documentation should be in an informal style and readable by the IT personnel. Guidance must be provided for every operational environment that the product supports as claimed in the ST. This guidance includes instructions to successfully install the TSF in that environment; and Instructions to manage the security of the TSF as a product and as a component of the larger operational environment. Guidance pertaining to particular security functionality is also provided; requirements on such guidance are contained in the ~~assurance~~ evaluation activities specified with each requirement.

AGD_OPE.1 Operational User Guidance (AGD_OPE.1)

Developer action elements:

AGD_OPE.1.1D
The developer shall provide operational user guidance.
Application Note: The
~~operation~~
operational user guidance does not have to be contained in a single document. Guidance to users, administrators and application developers can be spread among documents or web pages. Where appropriate, the guidance documentation is

expressed in the eXtensible Configuration Checklist Description Format (XCCDF) to support security automation. Rather than repeat information here, the developer should review the
~~assurance~~
evaluation activities for this component to ascertain the specifics of the guidance that the evaluator will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

Content and presentation elements:

AGD_OPE.1.
~~1C~~
2C
The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.
Application Note: User and administrator are to be considered in the definition of user role.
AGD_OPE.1.
~~2C~~
3C
The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.
AGD_OPE.1.
~~3C~~
4C
The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.
AGD_OPE.1.
~~4C~~
5C
The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.
AGD_OPE.1.
~~5C~~
6C
The operational user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences, and implications for maintaining secure operation.
AGD_OPE.1.
~~6C~~
7C
The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfill the security objectives for the operational environment as described in the ST.
AGD_OPE.1.
~~7C~~
8C
The operational user guidance shall be clear and reasonable.

Evaluator action elements:

AGD_OPE.1.
~~1E~~
9E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
~~Assurance~~
Evaluation Activity
~~➤~~
Guidance
Some of the contents of the operational guidance will be verified by the
~~assurance~~
evaluation activities in Section 5.1 Security Functional Requirements and evaluation of the TOE according to the [CEM]. The following additional information is also required.

If cryptographic functions are provided by the TOE, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

The documentation must describe the process for verifying updates to the TOE by verifying a digital signature – this may be done by the TOE or the underlying platform.

The evaluator shall verify that this process includes the following steps:

- *Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).*

- *Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the*

~~hash/~~

- *digital signature. The TOE will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.*

AGD_PRE.1 Preparative Procedures (AGD_PRE.1)

Developer action elements:

AGD_PRE.1.1D
The developer shall provide the TOE, including its preparative procedures.
Application Note: As with the operational guidance, the developer should look to the
~~assurance~~
evaluation activities to determine the required content with respect to preparative procedures.

Content and presentation elements:

AGD_PRE.1.
~~1C~~
2C
The preparative procedures shall describe all the steps necessary for secure acceptance of the deliveredTOE in accordance with the developer's delivery procedures.
AGD_PRE.1.
~~2C~~
3C
The preparative procedures shall describe all the steps necessary for secure installation of theTOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

Evaluator action elements:

AGD_PRE.1.
~~1E~~
4E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
AGD_PRE.1.
~~2E~~
5E
The evaluator shall apply the preparative procedures to confirm that theTOE can be prepared securely for operation.
~~Assurance~~
Evaluation Activity
~~➤~~
Guidance
As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support TOE functional requirements. The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms claimed for the TOE in the ST.

5.2.4 Class ALC: Life-cycle Support

At the assurance level provided for TOEs conformant to this PP, life-cycle support is limited to end-user-visible aspects of the life-cycle, rather than an examination of the TOE vendor's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it is a reflection on the information to be made available for evaluation at this assurance level.

ALC_CMC.1 Labeling of the TOE (ALC_CMC.1)

Developer action elements:

ALC_CMC.1.1D
The developer shall provide the TOE and a reference for the TOE.

Content and presentation elements:

ALC_CMC.1.
~~1C~~
2C

The
~~TOE~~
application shall be labeled with a unique reference.
Application Note: Unique reference information includes:

- Application Name
- Application Version
- Application Description
- Platform on which Application Runs
- Software Identification (SWID) tags, if available

Evaluator action elements:

ALC_CMC.1.
~~1E~~
3E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
~~Assurance~~
Evaluation Activity
➤
TSS
The evaluator shall check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST. Further, the evaluator shall check the AGD guidance and TOE samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the TOE, the evaluator shall examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

ALC_CMS.1 TOE CM Coverage (ALC_CMS.1)

Developer action elements:

ALC_CMS.1.1D
The developer shall provide a configuration list for the TOE.

Content and presentation elements:

ALC_CMS.1.
~~1C~~
2C
The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.
ALC_CMS.1.
~~2C~~
3C
The configuration list shall uniquely identify the configuration items.

Evaluator action elements:

ALC_CMS.1.
~~1E~~
4E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
~~Assurance~~
Evaluation Activity
➤
TSS
The "evaluation evidence required by the SARs" in this PP is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements. By ensuring that the TOE is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the
~~*assurance*~~
evaluation activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component. Life-cycle support is targeted aspects of the developer's life-cycle and instructions to providers of applications for the developer's devices, rather than an in-depth examination of the TSF manufacturer's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it's a reflection on the information to be made available for evaluation.
Guidance
The evaluator shall ensure that the developer has identified (in guidance documentation for application developers concerning the targeted platform) one or more development environments appropriate for use in developing applications for the developer's platform. For each of these development environments, the developer shall provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment(s) are invoked (e.g., compiler flags). The evaluator shall ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator shall ensure that the TSF is uniquely identified (with

respect to other products from the TSF vendor), and that documentation provided by the developer in association with the requirements in the ST is associated with the TSF using this unique identification.

ALC_TSU_EXT.1 Timely Security Updates

Developer action elements:

ALC_TSU_EXT.1.1D
The developer shall provide a description in the TSS of how timely security updates are made to theTOE.
Note: Application developers must support updates to their products for purposes of fixing security vulnerabilities.
ALC_TSU_EXT.1.2D
The developer shall provide a description in the TSS of how users are notified when updates change security properties or the configuration of the product.

Content and presentation elements:

ALC_TSU_EXT.1.
~~1C~~
3C
The description shall include the process for creating and deploying security updates for theTOE software.
ALC_TSU_EXT.1.
~~2C~~
4C
The description shall express the time window as the length of time, in days, between public disclosure of a vulnerability and the public availability of security updates to the TOE.
ALC_TSU_EXT.1.
~~3C~~
5C
The description shall include the mechanisms publicly available for reporting security issues pertaining to theTOE.
Note: The reporting mechanism could include web sites, email addresses, as well as a means to protect the sensitive nature of the report (e.g., public keys that could be used to encrypt the details of a proof-of-concept exploit).

Evaluator action elements:

ALC_TSU_EXT.1.
~~1E~~
6E
The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.
~~Assurance~~
Evaluation Activity
➤
TSS
The evaluator shall verify that the TSS contains a description of the timely security update process used by the developer to create and deploy security updates. The evaluator shall verify that this description addresses the entire application. The evaluator shall also verify that, in addition to the TOE developer's process, any third-party processes are also addressed in the description. The evaluator shall also verify that each mechanism for deployment of security updates is described.

The evaluator shall verify that, for each deployment mechanism described for the update process, the TSS lists a time between public disclosure of a vulnerability and public availability of the security update to the TOE patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator shall verify that this time is expressed in a number or range of days.

The evaluator shall verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the TOE. The evaluator shall verify that the description of this mechanism includes a method for protecting the report either using a public key for encrypting email or a trusted channel for a website.

5.2.5 Class ATE: Tests

Testing is specified for functional aspects of the system as well as aspects that take advantage of design or implementation weaknesses. The former is done through the ATE_IND family, while the latter is through the AVA_VAN family. At the assurance level specified in this PP, testing is based on advertised functionality and interfaces with dependency on the availability of design information. One of the primary outputs of the evaluation process is the test report as specified in the following requirements.

ATE_IND.1 Independent Testing – Conformance (ATE_IND.1)

Developer action elements:

ATE_IND.1.1D
The developer shall provide the TOE for testing.
Application Note: The developer must provide at least one product instance of theTOE for complete testing on at least platform regardless of equivalency. See the Equivalency Appendix for more details.

Content and presentation elements:

ATE_IND.1.
~~1C~~
2C
The TOE shall be suitable for testing.

Evaluator action elements:

ATE_IND.1.
~~1E~~
3E
The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.
ATE_IND.1.
~~2E~~
4E
The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.
Application Note: The evaluator shall test the application on the most current fully patched version of the platform.
~~Assurance~~
Evaluation Activity

Tests
The evaluator shall prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator shall determine the root cause of any application crashes and include that information in the report. The test plan covers all of the testing actions contained in the [CEM] and the body of this
~~*PP's Assurance Activities*~~
PP's evaluation activities.

While it is not necessary to have one test case per test listed in an
~~*Assurance Activity*~~
evaluation activity, the evaluator must document in the test plan that each applicable testing requirement in theST is covered. The test plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no
~~*affect*~~
effect; rationale must be provided. If all platforms claimed in theST are tested, then no rationale is necessary. The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform.

This also includes the configuration of the cryptographic engine to be used. The cryptographic algorithms implemented by this engine are those specified by this PP and used by the cryptographic protocols being evaluated (
~~*IPsec, TLS,*~~
e.g SSH). The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results.

The test report (which could just be an annotated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a "fail" and "pass" result (and the supporting details), and not just the "pass" result.

5.2.6 Class AVA: Vulnerability Assessment

For the current generation of this protection profile, the evaluation lab is expected to survey open sources to discover what vulnerabilities have been discovered in these types of products. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. Until penetration tools are created and uniformly distributed to the evaluation labs, the evaluator will not be expected to test for these vulnerabilities in the TOE. The labs will be expected to comment on the likelihood of these vulnerabilities given the documentation provided by the vendor. This information will be used in the development of penetration testing tools and for the development of future protection profiles.

AVA_VAN.1 Vulnerability Survey (AVA_VAN.1)

Developer action elements:

AVA_VAN.1.1D
The developer shall provide the TOE for testing.

Content and presentation elements:

AVA_VAN.1.
~~1C~~
2C
The
~~TOE~~
application shall be suitable for testing.
Application Note: Suitability for testing means not being obfuscated or packaged in such a way as to disrupt either static or dynamic analysis by the evaluator.

Evaluator action elements:

AVA_VAN.1.
~~1E~~
3E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
AVA_VAN.1.
~~2E~~
4E
The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.
Application Note: Public domain sources include the Common Vulnerabilities and Exposures (CVE) dictionary for publicly known vulnerabilities. Public domain sources also include sites which provide free checking of files for viruses.
AVA_VAN.1.
~~3E~~
5E
The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.
~~Assurance~~
Evaluation Activity
➤
Tests
The evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in similar applications with a particular focus on network protocols the application uses and document formats it parses. The evaluator shall also run a virus scanner with the most current virus definitions against the application files and verify that no files are flagged as malicious. The evaluator documents the sources consulted and the vulnerabilities found in the report.

For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

Appendix A

~~.~~

- Optional Requirements

As indicated in ~~Section 2~~the introduction to this PP, the baseline requirements (those that must be performed by the TOE) are contained in the body of this PP. ~~Additionally, there are~~ This Appendix contains three other types of optional requirements ~~specified in Appendix A, Appendix B, and Appendix C. The first type (in this Appendix) are requirements that can be included~~ that may be included in the ST, but are not required in order to conform to this PP. However, applied modules, packages and/or use cases may refine specific requirements as mandatory.

The first type (A.1 Strictly Optional Requirements) are strictly optional requirements that are independent of the TOE implementing any function. If the TOE fulfills any of these requirements or supports a certain functionality, the vendor is encouraged to included the SFRs in the ST, but are not required in order ~~for a TOE~~ to ~~claim conformance~~ conform to this PP.

The second type (~~in Appendix B~~ A.2 Objective Requirements) are ~~requirements based on selections in the body of the PP: if certain selections are made, then additional requirements in that appendix must be included. The third type (in Appendix C are components that are not required in order to conform to~~ objective requirements that describe security functionality not yet widely available in commercial technology. The requirements are not currently mandated in the body of this PP, but will be included in the baseline requirements in future versions of this PP~~, so adoption~~. Adoption by vendors is encouraged~~. Note that the ST author is responsible for ensuring that requirements that may be associated with those in Appendix A, Appendix B, and Appendix C but are not listed (e.g., FMT-type requirements) are also included in the ST. FCS_CKM.1(2)~~ and expected as soon as possible.

The third type (A.3 Implementation-Dependent 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.

A.1 Strictly Optional Requirements

A.1.1 Cryptographic Support (FCS)

FCS_CKM.1/2 Cryptographic Symmetric Key Generation

FCS_CKM.1.1
~~(~~
/2
~~)~~
The **application** shall generate **symmetric** cryptographic keys **using a Random Bit Generator as specified in FCS_RBG_EXT.1** and specified cryptographic key sizes [**selection**:

- *128 bit*,
- *256 bit*

].
Application Note: Symmetric keys may be used to generate keys along the key chain.
~~Assurance~~
Evaluation Activity
TSS
The evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked.

If the application is relying on random bit generation from the host platform, the evaluator shall verify the TSS includes the name/manufacturer of the external RBG and describes the function call and parameters used when calling the external DRBG function. If different external RBGs are used for different platforms, the evaluator shall verify the TSS identifies each RBG for each platform. Also, the evaluator shall verify the TSS includes a short description of the vendor's assumption for the amount of entropy seeding the external DRBG. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT or documentation available for the operational environment to determine that the key size being requested is identical to the key size and mode to be used for the encryption/decryption of the user data.
~~FCS_TLSC~~

A.2 Objective Requirements

A.2.1 Protection of the TSF (FPT)

FPT_API_EXT.2

~~TLS Client ProtocolFCS_TLSC~~

Use of Supported Services and APIs

FPT_API_EXT.2.1
The application
~~shall support mutual authentication using X.509v3 certificates.~~
[**selection**: *shall use platform-provided libraries*, *does not implement functionality*] for parsing [**assignment**: list of formats parsed that are included in the IANA MIME media types].
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.~~
~~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. B.~~
IANA MIME types are listed at http://www.iana.org/assignments/media-types and include many image, audio, video, and content file formats. This requirement does not apply if providing parsing services is the purpose of the application.
Evaluation Activity
TSS

The evaluator shall verify that the
~~AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates forTLS mutual authentication.~~

~~The evaluator shall also perform the following test:~~

- ~~**Test 1:** 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.~~

TSS lists the IANA MIME media types (as described by http://www.iana.org/assignments/media-types) for all formats the application processes and that it maps those formats to parsing services provided by the platform.

A.3 Implementation-Dependent Requirements

This PP does not define any implementation-dependent requirements.

Appendix B - Selection-Based Requirements

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this PP. There are additional requirements based on selections in the body of the PP: if certain selections are made, then additional requirements below ~~will need to~~ must be included.

B.1 Cryptographic Support (FCS)

FCS_RBG_EXT.2 Random Bit Generation from Application

This is a selection-based component. Its inclusion depends upon selection from FCS_RBG_EXT.1.1.
FCS_RBG_EXT.2.1
The application shall perform all deterministic random bit generation (DRBG) services in accordance with NIST Special Publication 800-90A using [**selection**: *Hash_DRBG (any)*, *HMAC_DRBG (any)*, *CTR_DRBG (AES)*]
~~.~~
~~***This requirement depends upon selection in FCS_RBG_EXT.1.1.***~~
Application Note: This requirement shall be included in
~~STs~~
STs in which *implement DRBG functionality* is chosen in FCS_RBG_EXT.1.1. The ST author should select the standard to which the RBG services comply (either SP 800-90A or FIPS 140-2 Annex C).

SP 800-90A contains three different methods of generating random numbers; each of these, in turn, depends on underlying cryptographic primitives (hash functions/ciphers). The ST author will select the function used (if SP 800-90A is selected), and include the specific underlying cryptographic primitives used in the requirement or in the TSS. While any of the identified hash functions (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CTR_DRBG are allowed.
~~Assurance~~
Evaluation Activity
➤
Tests
The evaluator shall perform the following tests, depending on the standard to which the RBG conforms.

Implementations Conforming to FIPS 140-2 Annex C.

The reference for the tests contained in this section is The Random Number Generator Validation System (RNGVS). The evaluators shall conduct the following two tests. Note that the "expected values" are produced by a reference implementation of the algorithm that is known to be correct. Proof of correctness is left to each Scheme.

- ***Test 1:*** *The evaluators shall perform a Variable Seed Test. The evaluators shall provide a set of 128 (Seed, DT) pairs to the TSF RBG function, each 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant for all 128 (Seed, DT) pairs. The DT value is incremented by 1 for each set. The seed values shall have no repeats within the set. The evaluators ensure that the values returned by the TSF match the expected values.*
- ***Test 2:*** *The evaluators shall perform a Monte Carlo Test. For this test, they supply an initial Seed and DT value to the TSF RBG function; each of these is 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluators then invoke the TSF RBG 10,000 times, with the DT value being incremented by 1 on each iteration, and the new seed for the subsequent iteration produced as specified in NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section E.3. The evaluators ensure that the 10,000th value produced matches the expected value.*

Implementations Conforming to NIST Special Publication 800-90A

- ***Test 1:*** *The evaluator shall perform 15 trials for the [RNG](#) implementation. If the [RNG](#) is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the [RNG](#) functionality.*

  If the [RNG](#) has prediction resistance enabled, each trial consists of (1) instantiate [DRBG](#), (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. "generate one block of random bits" means to generate random bits with number of returned bits equal to the Output Block Length (as defined in [NIST](#) [SP](#) 800-90A).

  If the [RNG](#) does not have prediction resistance, each trial consists of (1) instantiate [DRBG](#), (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

  The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

  Entropy input: *the length of the entropy input value must equal the seed length.*

  Nonce: *If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.*

  Personalization string: *The length of the personalization string must be less then or equal to seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.*

  Additional input: *the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.*

[FCS_RBG_EXT.2.2](#)
The deterministic [RBG](#) shall be seeded by an entropy source that accumulates entropy from a platform-based [DRBG](#) and [**selection**:

- *a software-based noise source*,
- *a hardware-based noise source*,
- *no other noise source*

] with a minimum of [**selection**:

- *128 bits*,
- *256 bits*

] of entropy at least equal to the greatest security strength (according to [NIST](#) [SP](#) 800-57) of the keys and hashes that it will generate.
~~***This requirement depends upon selection in [FCS_RBG_EXT.1.1](#).***~~
Application Note: This requirement shall be included in
~~[ST](#)s~~
STs in which *implement [DRBG](#) functionality* is chosen in [FCS_RBG_EXT.1.1](#). For the first selection in this requirement, the [ST](#) author selects 'software-based noise source' if any additional noise sources are used as input to the application's [DRBG](#). Note that the application must use the platform's [DRBG](#) to seed its [DRBG](#).

In the second selection in this requirement, the [ST](#) author selects the appropriate number of bits of entropy that corresponds to the greatest security strength of the algorithms included in the [ST](#). Security strength is defined in Tables 2 and 3 of [NIST](#) [SP](#) 800-57A. For example, if the implementation includes 2048-bit RSA (security strength of 112 bits) and [AES](#) 256 (security strength 256 bits), then the [ST](#) author would select 256 bits.
~~Assurance~~
[Evaluation Activity](#)

[TSS](#)
Documentation shall be produced - and the evaluator shall perform the activities - in accordance with [Appendix](#)
~~D and~~
[C - Entropy Documentation and Assessment](#) and the [Clarification to the Entropy Documentation and Assessment Annex](#)
Tests
In the future, specific statistical testing (in line with [NIST](#) [SP](#) 800-90B) will be required to verify the entropy estimates.

FCS_CKM

~~_EXT~~

.1/1 Cryptographic Asymmetric Key Generation

~~Services~~
~~FCS_CKM_EXT.1.1~~
~~The application shall [**selection**:~~

~~*generate no asymmetric cryptographic keys*,~~

~~*invoke platform-provided functionality for asymmetric key generation*,~~

~~*implement asymmetric key generation*~~

~~] . This requirement depends upon selection in FCS_TLSC~~

This is a selection-based component. Its inclusion depends upon selection from FCS_CKM_EXT.1.1.
~~**Application Note:** If *implement asymmetric key generation* or *invoke platform-provided functionality for asymmetric key generation* is chosen, then additional~~
FCS_CKM.1
~~(1) elements shall be included in the ST.~~
~~Assurance Activity~~
~~*The evaluator shall inspect the application and its developer documentation to determine if the application needs asymmetric key generation services. If not, the evaluator shall verify the **generate no asymmetric cryptographic keys** selection is present in the ST. Otherwise, the evaluation activities shall be performed as stated in the selection-based requirements.*~~

~~**FCS_CKM.1(1) Cryptographic Asymmetric Key Generation**~~

~~FCS_CKM.1.1(1)~~
~~The application shall~~
.1/1
The **application** shall [**selection**:

- *invoke platform-provided functionality*,
- *implement functionality*

] **to generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm** [**selection**:

- ***[RSA schemes]*** *using cryptographic key sizes of* ***[2048-bit or greater]*** *that meet the following*

~~: [**selection**:~~

- ***FIPS PUB 186-4,***

~~"Digital~~

- ***"Digital Signature Standard (DSS)***

~~"~~

- ***, Appendix B.3"***,

~~***ANSI X9.31-1998, Section 4.1***~~

~~],~~

- ***[ECC schemes]*** *using* ***[***

~~"NIST ***This requirement depends upon selection in FCS_CKM_EXT.1.1.***~~

- ***"NIST curves" P-256, P-384 and [selection: P-521 , no other curves]]*** *that meet the following:* ***[FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4]*** ,
- ***[FFC schemes]*** *using cryptographic key sizes of* ***[2048-bit or greater]*** *that meet the following:* ***[FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.1]*** ,

~~].~~
~~Assurance Activity~~

- *[FFC Schemes]* ***using Diffie-Hellman group 14*** *that meet the following:* ***RFC 3526, Section 3*** ,
- *[FFC Schemes]* ***using "safe-prime" groups*** *that meet the following:* ***NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"*** *and*

*[**selection**: RFC 3526, RFC 7919]*

].
Application Note: The ST author shall select all key generation schemes used for key establishment and entity authentication. When key generation is used for key establishment, the schemes in FCS_CKM.2.1 and selected cryptographic protocols must match the selection. When key generation is used for entity authentication, the public key is expected to be associated with an X.509v3 certificate.

If the TOE acts as a receiver in the RSA key establishment scheme, theTOE does not need to implement RSA key generation.
~~The ANSI X9.31-1998 option will be removed from the selection in a future publication of this document. Presently, the selection is not exclusively limited to the FIPS PUB 186-4 options in order to allow industry some further time to complete the transition to the modern FIPS PUB 186-4 standard.~~
Evaluation Activity
TSS

The evaluator shall ensure that the TSS identifies the key sizes supported by theTOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

*If the application **invokes platform-provided functionality for asymmetric key generation**, then the evaluator shall examine the TSS to verify that it describes how the key generation functionality is invoked.*

Guidance
The evaluator shall verify that the AGD guidance instructs the administrator how to configure theTOE to use the selected key generation scheme(s) and key size(s) for all uses defined in this PP.
~~If the application **invokes platform-provided functionality for asymmetric key generation**, then the evaluator shall examine the TSS to verify that it describes how the key generation functionality is invoked.~~
Tests

*If the application **implements asymmetric key generation**, then the following test activities shall be carried out.*

~~Assurance~~

Evaluation Activity Note: The following tests may require the developer to provide access to a developer environment that provides the evaluator with tools that are typically available to end-users of the application.

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by theTOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d. Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

1. *Random Primes:*
   - *Provable primes*
   - *Probable primes*
2. *Primes with Conditions:*
   - *Primes p1, p2, q1,q2, p and q shall all be provable primes*
   - *Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes*
   - *Primes p1, p2, q1,q2, p and q shall all be probable primes*

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of theTSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

If possible, the Random Probable primes method should also be verified against a known good implementation as described above. Otherwise, the evaluator shall have the TSF generate 10 keys pairs for each supported key length nlen and verify:

- *n =*

~~p*q~~

- *p·q,*
- *p and q are probably prime according to Miller-Rabin tests,*
- *GCD(p-1,e) = 1,*
- *GCD(q-1,e) = 1,*

~~2^16 <= e <= 2^256~~

- *$2^{16} \le e \le 2^{256}$ and e is an odd integer,*
- *|p-q| >*

~~2^(~~

- *$2^{nlen/2 - 100}$*

~~)~~

- *,*
- *p*

~~>= squareroot(2)*(2^(~~

- *$\ge 2^{nlen/2 -1}$*

~~))~~

- *$^{/2}$,*
- *q*

~~>= squareroot(2)*(2^(~~

- *$\ge 2^{nlen/2 -1}$*

~~))~~

- *$^{/2}$,*

~~2^~~

- *$2^{(nlen/2)} < d < LCM(p-1,q-1)$,*

~~e*d~~

- *$e \cdot d = 1 \bmod LCM(p-1,q-1)$.*

Key Generation for

~~ANSI X9.31-1998 RSA Schemes~~

~~If the TSF implements the ANSI X9.31-1998 scheme, the evaluator shall check to ensure that the TSS describes how the key-pairs are generated. In order to show that the TSF implementation complies with ANSI X9.31-1998, the evaluator shall ensure that the TSS contains the following information:~~

- ~~The TSS shall list all sections of the standard to which the TOE complies;~~
- ~~For each applicable section listed in the TSS, for all statements that are not "shall" (that is, "shall not", "should", and "should not") , if the TOE implements such options it shall be described in the TSS. If the included functionality is indicated as "shall not" or "should not" in the standard, the TSS shall provide a rationale for why this will not adversely affect the security policy implemented by the TOE;~~
- ~~For each applicable section of Appendix B, any omission of functionality related to "shall" or “should” statements shall be described.~~

~~Key Generation for~~

Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y. The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime *q and the field prime p:*

Cryptographic and Field Primes:

- *Primes q and p shall both be provable primes*
- *Primes q and field prime p shall both be probable primes*

and two ways to generate the cryptographic group generator g:

Cryptographic Group Generator:

- *Generator g constructed through a verifiable process*
- *Generator g constructed through an unverifiable process.*

The Key generation specifies 2 ways to generate the private key x: Private Key:

- *len(q) bit output of RBG where 1*

~~<=x <=~~

- *≤x ≤ q-1*
- *len(q) + 64 bit output of RBG, followed by a mod q-1 operation where*

~~1<= x<=q~~

- *1≤ x≤q-1.*

The security strength of the RBG must be at least that of the security offered by the FFC parameter set. To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set. For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

- *g*

~~!=~~

- *≠ 0,1*
- *q divides p-1*

~~g^q~~

- *g^q mod p = 1*

~~g^x~~

- *g^x mod p = y*

for each FFC parameter set and key pair.

Diffie-Hellman Group 14 and FFC Schemes using "safe-prime" groups

Testing for FFC Schemes using Diffie-Hellman group 14 and/or safe-prime groups is done as part of testing in CKM.2.1.

FCS_CKM.1/3 Password Conditioning

This is a selection-based component. Its inclusion depends upon selection from FCS_STO_EXT.1.1.
FCS_CKM.1.1/3
Refinement: A password/passphrase shall perform [Password-based Key Derivation Functions] in accordance with a specified cryptographic algorithm as specified in FCS_COP.1/4, with [assignment: positive integer of 1,000 or more] iterations, and output cryptographic key sizes [**selection**: *128*, *256*] that meet the following **[NIST SP 800-132]**.
FCS_CKM.1.2/3
The TSF shall generate salts using a RBG that meets FCS_RGB_EXT.1 and with entropy corresponding to the security strength selected for PBKDF in FCS_CKM.1.1/3
Application Note: This should be included if selected in FCS_STO_EXT.1

Conditioning can be performed using one of the identified hash functions or the process described in NIST SP 800-132; the method used is selected by the ST Author. SP 800-132 requires the use of a pseudo-random function (PRF) consisting of HMAC with an approved hash function. The ST author selects the hash function used, also includes the appropriate requirements for HMAC and the hash function.

Appendix A of SP 800-132 recommends setting the iteration count in order to increase the computation needed to derive a key from a password and, therefore, increase the workload of performing a password recovery attack. A significantly higher

value is recommended to ensure optimal security. This value is expected to increase to a minimum of 10,000 in a future iteration based on SP800-63.
Evaluation Activity
TSS
Support for PBKDF: The evaluator shall examine the password hierarchy TSS to ensure that the formation of all password based derived keys is described and that the key sizes match that described by the ST author. The evaluator shall check that the TSS describes the method by which the password/passphrase is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function. For the NIST SP 800-132-based conditioning of the password/passphrase, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements (FCS_COP.1.1/4). No explicit testing of the formation of the submask from the input password is required. FCS_CKM.1.1/3: The ST author shall provide a description in the TSS regarding the salt generation. The evaluator shall confirm that the salt is generated using an RBG described in FCS_RBG_EXT.1.

FCS_CKM.2 Cryptographic Key Establishment

This is a selection-based component. Its inclusion depends upon selection from FTP_DIT_EXT.1.1.
FCS_CKM.2.1
The application shall [**selection**: *invoke platform-provided functionality, implement functionality*] to perform cryptographic key establishment in accordance with a specified cryptographic key establishment method:

[**selection**:

- ***[RSA-based key establishment schemes]*** *that meets the following:* ***[NIST Special Publication 800-56B, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography"]*** ,

~~and [**selection**: ***This requirement depends upon selection in FCS_TLSC_EXT.1.1.***~~

- ***[RSA-based key establishment schemes]*** *that meet the following:* ***RSAES-PKCS1-v1_5 as specified in Section 7.2 of RFC 8017, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1"*** ,
- ***[Elliptic curve-based key establishment schemes]*** *that meets the following:* ***[NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"]*** ,
- ***[Finite field-based key establishment schemes]*** *that meets the following:* ***[NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"]*** ,

  ~~*No other schemes*~~

~~] .~~

- ***[Key establishment scheme using Diffie-Hellman group 14]*** *that meets the following:* ***RFC 3526, Section 3*** ,
- ***[FFC Schemes using "safe-prime" groups]*** *that meet the following:* ***'NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"*** *and [**selection**: RFC 3526, RFC 7919].*

].
Application Note: The ST author shall select all key establishment schemes used for the selected cryptographic protocols.
~~FCS_TLSC_EXT.1~~
TLS requires cipher suites that use RSA-based key establishment schemes.

The RSA-based key establishment schemes are described in Section 9 of NIST SP 800-56B; however, Section 9 relies on implementation of other sections in SP 800-56B. If the TOE acts as a receiver in the RSA key establishment scheme, the TOE does not need to implement RSA key generation.

The elliptic curves used for the key establishment scheme shall correlate with the curves specified in FCS_CKM.1.1
~~(~~
/1
~~)~~
.

The domain parameters used for the finite field-based key establishment scheme are specified by the key generation according to FCS_CKM.1.1
~~(~~
/1
~~)~~
.
~~Assurance~~
Evaluation Activity
➤
TSS
The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes

identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine theTSS to verify that it identifies the usage for each scheme.
Guidance
The evaluator shall verify that the AGD guidance instructs the administrator how to configure theTOE to use the selected key establishment scheme(s).
~~Assurance~~
Tests
Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Key Establishment Schemes

The evaluator shall verify the implementation of the key establishment schemes supported by the TOE using the applicable tests below.

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the DKM, the corresponding
~~TOE's~~
TOE's public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information (OtherInfo) and TOE id fields.

If the TOE does not use a KDF defined inSP 800-56A, the evaluator shall obtain only the public keys and the hashed value of the shared secret.

The evaluator shall verify the correctness of the
~~TSF's~~
TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the
~~TOE's~~
TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the OtherInfo andTOE id fields.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the OtherInfo field, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the
~~TOE's~~
TOE's static private key to assure theTOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters.

The evaluator shall compare the
~~*TOE's*~~
TOE's results with the results using a known good implementation verifying that theTOE detects these errors.

SP800-56B Key Establishment Schemes

The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

If the TOE acts as a sender, the following
~~*assurance*~~
evaluation activity shall be performed to ensure the proper operation of everyTOE supported combination of RSA-based key establishment scheme:

> *To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MacKey and MacTag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.*

If the TOE acts as a receiver, the following
~~*assurance*~~
evaluation activities shall be performed to ensure the proper operation of everyTOE supported combination of RSA-based key establishment scheme:

> *To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.*

The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance withNIST Special Publication 800-56B, the TOE must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEM-KWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

RSA-based key establishment

The evaluator shall verify the correctness of the TSF's implementation of RSAES-PKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses RSAES-PKCS1-v1_5.

Diffie-Hellman Group 14

The evaluator shall verify the correctness of the TSF's implementation of Diffie-Hellman group 14 by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses Diffie-Hellman group 14.

FFC Schemes using "safe-prime" groups

The evaluator shall verify the correctness of the TSF's implementation of safe-prime groups by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses.

FCS_COP.1

~~(~~

/1

~~)~~

Cryptographic Operation - Encryption/Decryption

This is a selection-based component. Its inclusion depends upon selection from FTP_DIT_EXT.1.1, FCS_STO_EXT.1.1.
FCS_COP.1.1
~~(~~
/1
~~)~~
The **application** shall perform *encryption/decryption* in accordance with a specified cryptographic algorithm [**selection**:

- *AES-CBC (as defined in NIST SP 800-38A) mode*

~~;~~

- ,

~~and [**selection**:~~

- *AES-GCM (as defined in NIST SP 800-38D) mode*,

~~*no other modes*~~

- *AES-XTS (as defined in NIST SP 800-38E) mode*

] and cryptographic key sizes
~~256-bit and~~
[**selection**: *128-bit*,
~~*no other key sizes*FCS_TLSC~~
256-bit] .
~~This requirement depends upon selection in~~
Application Note: This is dependent on implementing cryptographic functionality, as in FTP_DIT_EXT.1.
~~1, ***FCS_STO_EXT.1.1***.~~

~~**Application Note:**~~
For the first selection, the ST author should choose the mode or modes in which AES operates. For the second selection, the ST author should choose the key sizes that are supported by this functionality. 128-bit key size is required in order to comply with
~~FCS_TLSC_EXT.1 and FCS_CKM.1(1), if those are selected.~~

~~Assurance Activity~~
certain TLS implementations.

Evaluation Activity
Guidance
The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required modes and key sizes is present.
Tests
The evaluator shall perform all of the following tests for each algorithm implemented by the TSF and used to satisfy the requirements of this PP:

AES-CBC Known Answer Tests

There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

- *KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all- zeros key. To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.*
- *KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys. To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value*

as input and AES-CBC decryption.

- *KAT-3. To test the encrypt functionality ofAES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.*
- *KAT-4. To test the encrypt functionality ofAES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].*

To test the decrypt functionality ofAES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i <= 10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation. AES-CBC Monte Carlo Tests The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3- tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
 if i == 1:
      CT[1] = AES-CBC-Encrypt(Key, IV, PT)
      PT = IV
 else:
   CT[i] = AES-CBC-Encrypt(Key, PT)
   PT = CT[i-1]
```

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-GCM Monte Carlo Tests

The evaluator shall test the authenticated encrypt functionality ofAES-GCM for each combination of the following input parameter lengths:

- 128 bit and 256 bit keys
- Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.
- Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.
- Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

AES-XTS Tests

The evaluator shall test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths:

256 bit (for AES-128) and 512 bit (for AES-256) keys

Three data unit (i.e., plaintext) lengths. One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 216 bits, whichever is smaller.

Using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS-AES decrypt.

FCS_COP.1

~~(~~

/2

~~)~~

Cryptographic Operation - Hashing

This is a selection-based component. Its inclusion depends upon selection from FTP_DIT_EXT.1.1.
FCS_COP.1.1
~~(~~
/2
~~)~~
The **application** shall perform *cryptographic hashing* services in accordance with a specified cryptographic algorithm [**selection**:

- *SHA-1*,
- *SHA-256*,
- *SHA-384*,
- *SHA-512*,
- *no other*

~~algorithms~~

-

] and message digest sizes [**selection**:

- *160*,
- *256*,
- *384*,
- *512*,
- *no other*

~~message digest sizesFCS_TLSC~~

-

] bits that meet the following: FIPS Pub 180-4.
~~This requirement depends upon selection in~~
Application Note: This is dependent on implementing cryptographic functionality, as in FTP_DIT_EXT.1.
~~1.~~

~~**Application Note:**~~
Per NIST SP 800-131A, SHA-1 for generating digital signatures is no longer allowed, and SHA-1 for verification of digital

signatures is strongly discouraged as there may be risk in accepting these signatures.

SHA-1 is currently
~~required~~
included in order to comply with
~~FCS_TLSC_EXT.1. If FCS_TLSC_EXT.1.1~~
the TLS. If the TLS package is included in the ST, the hashing algorithms selection for FCS_COP.1(2) must match the hashing algorithms used in the mandatory and selected cipher suites of
~~FCS_TLSC_EXT.1.1.~~
the TLS package. Vendors are strongly encouraged to implement updated protocols that support the SHA-2 family; until updated protocols are supported, this PP allows support for SHA-1 implementations in compliance with SP 800-131A.

The intent of this requirement is to specify the hashing function. The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used (for example, SHA 256 for 128-bit keys).
~~Assurance~~
Evaluation Activity
➤
TSS
The evaluator shall check that the association of the hash function with other application cryptographic functions (for example, the digital signature verification function) is documented in the TSS.
Tests
The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF hashes only messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs. The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

- ***Test 1:*** *Short Messages Test - Bit oriented Mode The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*
- ***Test 2:*** *Short Messages Test - Byte oriented Mode The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*
- ***Test 3:*** *Selected Long Messages Test - Bit oriented Mode The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where 1 ≤ i ≤ m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*
- ***Test 4:*** *Selected Long Messages Test - Byte oriented Mode The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where 1 ≤ i ≤ m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*
- ***Test 5:*** *Pseudorandomly Generated Messages Test This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.*

FCS_COP.1

~~(~~

/3

~~)~~

Cryptographic Operation - Signing

This is a selection-based component. Its inclusion depends upon selection from FTP_DIT_EXT.1.1.
FCS_COP.1.1
~~(~~
/3
~~)~~
The **application** shall perform *cryptographic signature services (generation and verification)* in accordance with a specified

cryptographic algorithm [**selection**:

- ***RSA schemes*** *using cryptographic key sizes of 2048-bit or greater that meet the following:FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 4* ,
- ***ECDSA schemes*** *using*

~~"NIST FCS_TLSC~~

- *"NIST curves" P-256, P-384 and [**selection**: P-521, no other curves] that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5*

] .
~~This requirement depends upon selection in~~
Application Note: This is dependent on implementing cryptographic functionality, as in FTP_DIT_EXT.1.
~~1.~~

~~**Application Note:**~~
The ST Author should choose the algorithm implemented to perform digital signatures; if more than one algorithm is available, this requirement should be iterated to specify the functionality. For the algorithm chosen, the ST author should make the appropriate assignments/selections to specify the parameters that are implemented for that algorithm.
~~RSA signature generation and verification is currently required in order to comply withFCS_TLSC_EXT.1. Assurance Activity~~ ➤
Evaluation Activity
The evaluator shall perform the following activities based on the selections in theST.
Tests
The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

ECDSA Algorithm Tests

- ***Test 1:*** *ECDSA FIPS 186-4 Signature Generation Test. For each supportedNIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.*
- ***Test 2:*** *ECDSA FIPS 186-4 Signature Verification Test. For each supportedNIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.*

RSA Signature Algorithm Tests

- ***Test 1:*** *Signature Generation Test. The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages. The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.*
- ***Test 2:*** *Signature Verification Test. The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys, e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.*

FCS_COP.1

~~(~~

/4

~~)~~

Cryptographic Operation - Keyed-Hash Message Authentication

This is a selection-based component. Its inclusion depends upon selection fromFTP_DIT_EXT.1.1.
FCS_COP.1.1
~~(~~
/4
~~)FCS_TLSC~~
The **application** shall perform *keyed-hash message authentication* in accordance with a specified cryptographic algorithm

- HMAC-SHA-256

and [**selection**:

- *SHA-1*,
- *SHA-384*,
- *SHA-512*,
- *no other algorithms*

] with key sizes [**assignment**: key size (in bits) used in HMAC] and message digest sizes 256 and [**selection**: *160*, *384*, *512*, *no other size*] bits that meet the following: FIPS Pub 198-1 *The Keyed-Hash Message Authentication Code* and FIPS Pub 180-4 *Secure Hash Standard*.
~~This requirement depends upon selection in~~
Application Note: This is dependent on implementing cryptographic functionality, as in FTP_DIT_EXT.1.
~~1.~~

~~Application Note:~~
The intent of this requirement is to specify the keyed-hash message authentication function used for key establishment purposes for the various cryptographic protocols used by the application (e.g., trusted channel). The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1
~~(~~
/1
~~)~~
.
~~HMAC-SHA256 is required in order to comply with the required cipher suites in FCS_TLSC_EXT.1. Assurance Activity~~
Evaluation Activity
The evaluator shall perform the following activities based on the selections in the ST.
Tests
For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and IV using a known-good implementation.

FCS_

~~TLSC~~

HTTPS_EXT.1

~~TLS Client Protocol~~
~~FCS_TLSC_EXT.1.1~~
~~The application shall [**selection**: *invoke platform-provided TLS 1.2, implement TLS 1.2 (RFC 5246)*] supporting the following cipher suites:~~

~~Mandatory Cipher Suites: TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246~~

~~Optional Cipher Suites: [**selection**:~~

~~*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_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*,~~

~~*TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246*,~~

~~*TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246*,~~

~~*no other cipher suite*~~

~~] . This requirement depends upon selection in~~

HTTPS Protocol

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

~~**Application Note:** The cipher suites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional cipher suites that are supported; if there are no cipher suites supported other than the mandatory suites, then "None" should be selected. It is necessary to limit the cipher suites that can be used in an evaluated configuration administratively on the server in the test environment. The Suite B algorithms listed above (RFC 6460) are the preferred algorithms for implementation. TLS_RSA_WITH_AES_128_CBC_SHA is required in order to ensure compliance with RFC 5246.~~

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

~~If any cipher suites are selected using ECDHE, then FCS_TLSC_EXT.4 is required.~~

~~If *implement TLS 1.2 (RFC 5246)* is selected, then FCS_CKM.2, FCS_CKM_EXT.1, FCS_COP.1(1), FCS_COP.1(2), FCS_COP.1(3), and FCS_COP.1(4) are required.~~

~~Assurance Activity~~

~~The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the cipher suites supported are specified. The evaluator shall check the TSS to ensure that the cipher suites specified include 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. The evaluator shall also perform the following tests:~~

- ~~**Test 1:** The evaluator shall establish a TLS connection using each of the cipher suites 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 EAP session. It is sufficient to observe the successful negotiation of a cipher suite 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 cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).~~

~~**Test 2:**~~

FCS_HTTPS_EXT.1.1

The application shall implement the HTTPS protocol that complies with RFC 2818.

Evaluation Activity

Tests

The evaluator shall attempt to establish

~~the~~

an HTTPS 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 does not match the server-selected cipher suite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite or send a RSA certificate while using one of the ECDSA cipher suites.) 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 cipher suite and verify that the client denies the connection.~~
- ~~**Test 5:** The evaluator shall perform the following modifications to the traffic:~~

  - ~~**Test 5.1:** 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.~~
  - ~~**Test 5.2:** 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 cipher suite) or that the server denies the client's Finished handshake message.~~
  - ~~**Test 5.3:** Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.~~
  - ~~**Test 5.4:** 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.~~
  - ~~**Test 5.5:** 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.~~
  - ~~**Test 5.6:** Send an garbled message 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 application shall verify that the presented identifier matches the reference identifier according to RFC 6125.~~

~~***This requirement depends upon selection in FTP_DIT_EXT.1.1.***~~

~~**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 application service. Based on a singular reference identifier's source domain and application service type (e.g. HTTP, SIP, LDAP), the client establishes all reference identifiers which are acceptable, such as a Common Name for the Subject Name field of the certificate and a (case-insensitive) DNS name, URI name, and Service Name for the Subject Alternative Name~~

~~field. The client then compares this list of all acceptable reference identifiers to the presented identifiers in the 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 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:~~*
    - ***~~Test 5.1:~~*** *~~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.~~*
    - ***~~Test 5.2:~~*** *~~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.~~*
    - ***~~Test 5.3:~~*** *~~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 URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.~~*
- ***~~Test 7:~~*** *~~[conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.~~*

~~FCS_TLSC_EXT.1.3~~

~~The application shall establish a trusted channel only if the peer certificate is valid.~~

~~This requirement depends upon selection in FTP_DIT_EXT.1.1.~~

~~**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.~~

~~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 overTLS. This element addresses non-HTTPS TLS connections.~~

~~Assurance Activity~~

~~The evaluator shall use TLS as a function to verify that the validation rules inFIA_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.4 TLS Client Protocol

FCS_TLSC_EXT.4.1
The application shall present the supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [**selection**: *secp256r1*, *secp384r1*, *secp521r1*] and no other curves.
This requirement depends upon selection in FCS_TLSC_EXT.1.1, FCS_TLSS_EXT.1.1.
Application Note: 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.2. This extension is required for clients supporting Elliptic Curve cipher suites.
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 perform an ECDHE key exchange message in the TLS connection using a non-supported ECDHE curve (for example, P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.*

FCS_TLSS_EXT.1 TLS Server Protocol

FCS_TLSS_EXT.1.1
The application shall [**selection**: *invoke platform-provided TLS 1.2*, implement TLS 1.2 (RFC 5246)] supporting the following cipher suites:

Mandatory Cipher Suites: TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246

Optional Cipher Suites: [**selection**:

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_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,

TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,

TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,

no other cipher suite

].
This requirement depends upon selection in FTP_DIT_EXT.1.1.
Application Note: The cipher suites to be tested in the evaluated configuration are limited by this requirement. TheST author should select the optional cipher suites that are supported; if there are no cipher suites supported other than the mandatory suites, then "None" should be selected. It is necessary to limit the cipher suites that can be used in an evaluated configuration administratively on the server in the test environment. The Suite B algorithms listed above (RFC 6460) are the preferred algorithms for implementation. TLS_RSA_WITH_AES_128_CBC_SHA is required in order to ensure compliance with RFC 5246.

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

If any cipher suites are selected using ECDHE, then FCS_TLSC_EXT.4 is required.

If *implement TLS 1.2 (RFC 5246)* is selected, then FCS_CKM.2.1, FCS_COP.1.1(1), FCS_COP.1.1(2), FCS_COP.1.1(3), and FCS_COP.1.1(4) are required.
Assurance Activity

The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the cipher suites supported are specified. The evaluator shall check the TSS to ensure that the cipher suites specified include 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 theTSS. The evaluator shall also perform the following tests:

- ***Test 1:*** *The evaluator shall establish a TLS connection using each of the cipher suites 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 EAP session. It is sufficient to observe the successful negotiation of a cipher suite 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 cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).*
- ***Test 2:*** *The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.*
- ***Test 3:*** *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 application disconnects after receiving the key exchange message.*
- ***Test 4:*** *The evaluator shall perform the following modifications to the traffic:*
  - ***Test 4.1:*** *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.*
  - ***Test 4.2:*** *Modify at least one 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.*
  - ***Test 4.3:*** *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.*
  - ***Test 4.4:*** *Modify a byte in the Clint Finished handshake message, and verify that the server rejects the connection and does not send any application data.*
  - ***Test 4.5:*** *After generating a fatal alert by sending a Finished message from the client before the client send a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection.*
  - ***Test 4.6:*** *Send an garbled message 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 application shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1, and [**selection**: *TLS 1.2, none*] .
This requirement depends upon selection in FTP_DIT_EXT.1.1.
Application Note: All SSL versions and TLS 1.0 and 1.1 are denied. Any TLS version 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 TLS versions, and any configuration necessary to meet the requirement must be contained in the AGD guidance.

- ***Test 1:*** *The evaluator shall send a Client Hello requesting a connection with version SSL 2.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 3.0, TLS 1.0, TLS 1.1, and TLS 1.2 if it was selected.*

FCS_TLSS_EXT.1.3
The application shall generate key establishment parameters using RSA with size 2048 bits and [**selection**: *3072 bits, 4096 bits, no other sizes*] and [**selection**: *over NIST curves [**selection**: secp256r, secp384r] and no other curves, Diffe-Hellman parameters of size 2048 and [**selection**: 3072 bits, no other size] , no other*]
This requirement depends upon selection in FTP_DIT_EXT.1.1.
Application Note: If the ST lists a DHE ciphersuite in FCS_TLSS_EXT.1.1, the ST must include the Diffie-Hellman selection in the requirement
Assurance Activity

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

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

- ***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.1.4
The application shall support mutual authentication of TLS clients using X.509v3 certificates.
This requirement depends upon selection in FTP_DIT_EXT.1.1.
FCS_TLSS_EXT.1.5
The application shall not establish a trusted channel if the peer certificate is invalid.
This requirement depends upon selection in FTP_DIT_EXT.1.1.

~~**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.~~
with a webserver, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS.
FCS_HTTPS_EXT.1.2
The application shall implement HTTPS using TLS as defined in the TLS package.
Evaluation Activity
Tests
Other tests are performed in conjunction with the TLS package.
FCS_HTTPS_EXT.1.3
The application shall [**selection**: *not establish the application-initiated connection* , *notify the user and not establish the user-initiated connection* , *notify the user and request authorization to establish the user-initiated connection*] if the peer certificate is deemed invalid.
Application Note: Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280.
Evaluation Activity
Tests
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.~~
- ~~**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~~
, and the evaluator shall perform the following test:

- ***Test 1:*** *The evaluator shall demonstrate that using a certificate without a valid certification path results in the*

~~function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. 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 perform the following modifications to the traffic: a) 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. b) Configure the server to require mutual authentication and then modify a byte in the client's Certificate Verify handshake message. The evaluator shall verify that the server rejects the connection.~~

~~FCS_TLSS_EXT.1.6~~
~~The application 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.~~
~~***This requirement depends upon selection in FTP_DIT_EXT.1.1.***~~
~~**Application Note:** 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~~
~~*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.*~~

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

~~**FCS_DTLS_EXT.1 DTLS Implementation**~~

~~FCS_DTLS_EXT.1.1~~
~~The application shall implement the DTLS protocol in accordance with DTLS 1.2 (RFC 6347).~~
~~***This requirement depends upon selection in FTP_DIT_EXT.1.1.***~~

~~Assurance Activity~~

- ~~**Test 1:** The evaluator shall attempt to establish a connection with a DTLS server, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as DTLS.~~

~~Other tests are performed in conjunction with the Assurance Activity listed for FCS_TLSC_EXT.1.~~

~~FCS_DTLS_EXT.1.2~~

~~The application shall implement the requirements in TLS (FCS_TLSC_EXT.1) for the DTLS implementation, except where variations are allowed according to DTLS 1.2 (RFC 6347).~~

~~***This requirement depends upon selection in FTP_DIT_EXT.1.1.***~~

~~**Application Note:** Differences between DTLS 1.2 and TLS 1.2 are outlined in RFC 6347; otherwise the protocols are the same. In particular, for the applicable security characteristics defined for the TSF, the two protocols do not differ. Therefore, all application notes and assurance activities that are listed for TLS apply to the DTLS implementation.~~

~~Assurance Activity~~

~~*The evaluator shall perform the assurance activities listed for FCS_TLSC_EXT.1.*~~

~~FCS_DTLS_EXT.1.3~~

~~The application shall not establish a trusted communication channel if the peer certificate is deemed invalid.~~

~~***This requirement depends upon selection in FTP_DIT_EXT.1.1.***~~

~~**Application Note:** Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280.~~

~~Assurance Activity~~

~~Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1, and the evaluator shall perform the following test. **Test 1:** The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing.~~

- *selected action in the SFR. If "notify the user" is selected in theSFR, then the evaluator shall also determine that the user is notified of the certificate validation failure. Using the administrative guidance, the evaluator shall then load a certificate or certificates to the Trust Anchor Database 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.~~

~~FCS_HTTPS_EXT.1 HTTPS Protocol~~

~~FCS_HTTPS_EXT.1.1~~

~~The application shall implement the HTTPS protocol that complies with RFC 2818.~~

~~***This requirement depends upon selection in FTP_DIT_EXT.1.1.***~~

~~Assurance Activity~~

~~*The evaluator shall attempt to establish an HTTPS connection with a webserver, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS.*~~

~~FCS_HTTPS_EXT.1.2~~

~~The application shall implement HTTPS using TLS (FCS_TLSC_EXT.1).~~

~~***This requirement depends upon selection in FTP_DIT_EXT.1.1.***~~

~~Assurance Activity~~

~~*Other tests are performed in conjunction with FCS_TLSC_EXT.1.*~~

~~FCS_HTTPS_EXT.1.3~~

~~The application shall notify the user and [**selection**: *not establish the connection* , *request application authorization to establish the connection* , *no other action*] if the peer certificate is deemed invalid.~~

~~***This requirement depends upon selection in FTP_DIT_EXT.1.1.***~~

~~**Application Note:** Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280.~~

~~Assurance Activity~~

~~Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1, and the evaluator shall perform the following test: **Test 1:** The evaluator shall demonstrate that using~~

- *again, using a certificate without a valid certification path results in*

~~an application notification. Using the administrative guidance, the evaluator shall then load a certificate or certificates to the Trust Anchor Database 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 application~~

- *the selected action in the SFR, and if "notify the user" was selected in theSFR, the user is notified of the validation failure.*

B.2 Identification and Authentication (FIA)

FIA_X509_EXT.1 X.509 Certificate Validation

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

FIA_X509_EXT.1.1

The application shall [**selection**: *invoked platform-provided functionality* , *implement functionality*] to validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted CA certificate.
- The application shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates.
- The application shall validate the revocation status of the certificate using [**selection**: *the Online Certificate Status Protocol (OCSP) as specified in RFC 2560* , *a Certificate Revocation List (CRL) as specified in RFC 5280 Section 6.3* , *a Certificate Revocation List (CRL) as specified in RFC 5759* , *an OCSP TLS Status Request Extension (i.e., OCSP stapling) as specified in RFC 6066*] .
- The application shall validate the extendedKeyUsage field according to the following rules:
  - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field.
  - S/MIME certificates presented for email encryption and signature shall have the Email Protection purpose (id-kp 4 with OID 1.3.6.1.5.5.7.3.4) in the extendedKeyUsage field.
  - OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the extendedKeyUsage field.
  - Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the extendedKeyUsage field.

~~***This requirement depends upon selection in FTP_DIT_EXT.1.1.***~~

Application Note: FIA_X509_EXT.1.1 lists the rules for validating certificates. The ST author shall select whether revocation status is verified using OCSP or CRLs. FIA_X509_EXT.2 requires that certificates are used for HTTPS, TLS and DTLS; this use requires that the extendedKeyUsage rules are verified.

Regardless of the selection of *implement functionality* or *invoke platform-provided functionality*, the validation is expected to end in a trusted root CA certificate in a root store managed by the platform.

~~Assurance~~

Evaluation Activity

TSS

The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

Tests

The tests described must be performed in conjunction with the other certificate services ~~*assurance*~~ *evaluation activities, including the functions in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. If the application supports chains of length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.*

- ***Test 1:*** *The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator shall then delete one of the certificates, and show that the function fails.*
- ***Test 2:*** *The evaluator shall demonstrate that validating an expired certificate results in the function failing.*
- ***Test 3:*** *The evaluator shall test that the TOE can properly handle revoked certificates-–conditional on whether CRL, OCSP, or OCSP Stapling is selected; if multiple methods are selected, then the following tests shall be performed for each method:*
  - *The evaluator shall test revocation of the node certificate.*
  - *The evaluator shall also test revocation of an intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA), if intermediate CA certificates are supported.*

  The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.
- ***Test 4:*** *If OCSP is selected, the evaluator shall configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set, and verify that validation of the CRL fails.*
- ***Test 5:*** *The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)*
- ***Test 6:*** *The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)*
- ***Test 7:*** *The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)*

FIA_X509_EXT.1.2

The application shall treat a certificate as a CA certificate only if the basicConstraints extension is present and the CA flag is set to TRUE.

~~***This requirement depends upon selection in FTP_DIT_EXT.1.1.***~~

Application Note: This requirement applies to certificates that are used and processed by the TSF and restricts the certificates that may be added as trusted CA certificates.
~~Assurance~~
Evaluation Activity

Tests
The tests described must be performed in conjunction with the other certificate services
~~assurance~~
evaluation activities, including the functions in FIA_X509_EXT.2.1. If the application supports chains of length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

- ***Test 1:*** *The evaluator shall*

~~construct a certificate path, such~~

- *ensure that the certificate of*

~~the CA issuing the TOE's certificate~~

- *at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator shall confirm that validation of the certificate path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate without the basicConstraints extension to the TOE's trust store.*
- ***Test 2:*** *The evaluator shall*

~~construct a certificate path, such~~

- *ensure that the certificate of*

~~the CA issuing the TOE's certificate~~

- *at least one of the CAs in the chain has the CA flag in the basicConstraints extension not set (or set to FALSE). The evaluator shall confirm that validation of the certificate path fails*

~~. **Test 3:** The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the CA flag~~

- *(i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate with the CA flag not set (or set to FALSE) in the basicConstraints extension*

~~set~~

- *to*

~~TRUE. The validation of the certificate path succeeds~~

- *the TOE's trust store.*

FIA_X509_EXT.2 X.509 Certificate Authentication

This is a selection-based component. Its inclusion depends upon selection from FTP_DIT_EXT.1.1.
FIA_X509_EXT.2.1
The application shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [**selection**: *HTTPS*, *TLS*, *DTLS*, *SSH*, *IPsec*].
~~***This requirement depends upon selection in FTP_DIT_EXT.1.1.***~~
Application Note: The ST author's selection shall match the selection in FTP_DIT_EXT.1.1.
FIA_X509_EXT.2.2
When the application cannot establish a connection to determine the validity of a certificate, the application shall [**selection**: *allow the administrator to choose whether to accept the certificate in these cases* , *accept the certificate* , *not accept the certificate*].
~~***This requirement depends upon selection in FTP_DIT_EXT.1.1.***~~
Application Note: Often a connection must be established to perform a verification of the revocation status of a certificate - either to download a CRL or to perform OCSP. The selection is used to describe the behavior in the event that such a connection cannot be established (for example, due to a network error). If the TOE has determined the certificate valid according to all other rules in FIA_X509_EXT.1, the behavior indicated in the selection shall determine the validity. The TOE must not accept the certificate if it fails any of the other validation rules in FIA_X509_EXT.1.
~~Assurance~~
Evaluation Activity

TSS
The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the

certificates.

The evaluator shall examine the [TSS](#) to confirm that it describes the behavior of the[TOE](#) when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.
Tests
The evaluator shall perform the following test for each trusted channel:

- ***Test 1:*** *The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE [IT](#) entity. The evaluator shall then manipulate the environment so that the [TOE](#) is unable to verify the validity of the certificate, and observe that the action selected in [FIA_X509_EXT.2.2](#) is performed. If the selected action is administrator-configurable, then the evaluator shall follow the operational guidance to determine that all supported administrator-configurable options behave in their documented manner.*
- ***Test 2:*** *The evaluator shall demonstrate that an invalid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE [IT](#) entity cannot be accepted.*

~~C. Objective Requirements~~

~~This Annex includes requirements that specify security functionality which also addresses threats. The requirements are not currently mandated in the body of this PP as they describe security functionality not yet widely available in commercial technology. However, these requirements may be included in the ST such that the TOE is still conformant to this PP, and it is expected that they be included as soon as possible.~~

~~**FCS_TLSC_EXT.3 TLS Client Protocol**~~

~~FCS_TLSC_EXT.3~~

B.3 Protection of the TSF (FPT)

FPT_TUD_EXT.2 Integrity for Installation and Update

This is a selection-based component. Its inclusion depends upon selection from[FPT_TUD_EXT.1.5](#).
[FPT_TUD_EXT.2.1](#)
The application 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_algorithm extension is only supported by TLS 1.2.~~
~~Assurance Activity~~
~~*The evaluator shall verify that [TSS](#) describes the signature_algorithm extension and whether the required behavior is performed by default or may be configured. If the [TSS](#) indicates that the signature_algorithm extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the signature_algorithm extension.*~~

~~*The evaluator shall also perform the following test:*~~

- ~~***Test 1:*** *The evaluator shall configure the server to send a certificate in the TLS connection that is not supported according to the Client's HashAlgorithm enumeration within the signature_algorithms extension (for example, send a certificate with a SHA-1 signature). The evaluator shall verify that the [TOE](#) disconnects after receiving the server's Certificate handshake message.*~~

~~**FPT_API_EXT.2 Use of Supported Services and APIs**~~

~~[FPT_API_EXT.2.1](#)~~
~~The application [**selection**: *shall use platform-provided libraries*, *does not implement functionality*] for parsing [**assignment**: list of formats parsed that are included in the IANA MIME media types] .~~
~~**Application Note:** The IANA MIME types are listed at http://www.iana.org/assignments/media-types and include many image, audio, video, and content file formats. This requirement does not apply if providing parsing services is the purpose of the application.~~
~~Assurance Activity~~
~~*The evaluator shall verify that the TSS lists the IANA MIME media types (as described by http://www.iana.org/assignments/media-types) for all formats the application processes and that it maps those formats to parsing services provided by the platform.*~~

~~**FPT_IDV_EXT.1 Software Identification and Versions**~~

~~FPT_IDV_EXT.1.1~~
~~The application shall include SWID tags that comply with the minimum requirements forSWID tag from ISO/IEC 19770-2:2009 standard.~~
~~**Application Note:** Valid SWID tags must contain a SoftwareIdentity element and an Entity element as defined in the ISO/IEC 19770-2:2009 standard. SWID tags must be stored with a .swidtag file extensions as defined in the ISO/IEC 19770-2:2009.~~
~~Assurance Activity~~
~~*The evaluator shall install the application, then check for the existence of SWID tags in a .swidtag file. The evaluator shall open the file and verify that is contains at least a SoftwareIdentity element and an Entity element.*~~
~~D.~~
be distributed using the format of the platform-supported package manager.
Evaluation Activity
Tests
The evaluator shall verify that application updates are distributed in the format supported by the platform. This varies per platform:
For Android: *The evaluator shall ensure that the application is packaged in the Android application package (APK) format.*
For Windows: *The evaluator shall ensure that the application is packaged in the standard Windows Installer (.MSI) format, the Windows Application Software (.EXE) format signed using the Microsoft Authenticode process, or the Windows Universal Application package (.APPX) format. See https://msdn.microsoft.com/en-us/library/ms537364(v=vs.85).aspx for details regarding Authenticode signing.*
For iOS: *The evaluator shall ensure that the application is packaged in the IPA format.*
For Linux: *The evaluator shall ensure that the application is packaged in the format of the package management infrastructure of the chosen distribution. For example, applications running on Red Hat and Red Hat derivatives shall be packaged in RPM format. Applications running on Debian and Debian derivatives shall be packaged inDEB format.*
For Solaris: *The evaluator shall ensure that the application is packaged in the PKG format.*
For macOS: *The evaluator shall ensure that application is packaged in the DMG format, the PKG format, or the MPKG format.*
FPT_TUD_EXT.2.2
The application shall be packaged such that its removal results in the deletion of all traces of the application, with the exception of configuration settings, output files, and audit/log events.
Application Note: Applications software bundled with the system/firmware image are not subject to this requirement if the user is unable to remove the application through means provided by the OS.
Evaluation Activity
Tests
For Android: *The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).*
For iOS: *The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).*
For All Other Platforms: *The evaluator shall record the path of every file on the entire filesystem prior to installation of the application, and then install and run the application. Afterwards, the evaluator shall then uninstall the application, and compare the resulting filesystem to the initial record to verify that no files, other than configuration, output, and audit/log files, have been added to the filesystem.*

Appendix C - Entropy Documentation and Assessment

This appendix describes the required supplementary information for the entropy source used by the TOE.
The documentation of the entropy source should be detailed enough that, after reading, the evaluator will thoroughly understand the entropy source and why it can be relied upon to provide sufficient entropy. This documentation should include multiple detailed sections: design description, entropy justification, operating conditions, and health testing. This documentation is not required to be part of the TSS.

~~D~~

C.1 Design Description

Documentation shall include the design of the entropy source as a whole, including the interaction of all entropy source components. Any information that can be shared regarding the design should also be included for any third-party entropy sources that are included in the product.
The documentation will describe the operation of the entropy source to include, how entropy is produced, and how unprocessed (raw) data can be obtained from within the entropy source for testing purposes. The documentation should walk through the entropy source design indicating where the entropy comes from, where the entropy output is passed next, any post-processing of the raw outputs (hash, XOR, etc.), if/where it is stored, and finally, how it is output from the entropy source. Any conditions placed on the process (e.g., blocking) should also be described in the entropy source design. Diagrams and examples are encouraged.
This design must also include a description of the content of the security boundary of the entropy source and a description of how the security boundary ensures that an adversary outside the boundary cannot affect the entropy rate.
If implemented, the design description shall include a description of how third-party applications can add entropy to the RBG. A description of any RBG state saving between power-off and power-on shall be included.

~~D~~

C.2 Entropy Justification

There should be a technical argument for where the unpredictability in the source comes from and why there is confidence in the entropy source delivering sufficient entropy for the uses made of the RBG output (by this particular TOE). This argument will include a description of the expected min-entropy rate (i.e. the minimum entropy (in bits) per bit or byte of source data) and explain that sufficient entropy is going into the TOE randomizer seeding process. This discussion will be part of a justification for why the entropy source can be relied upon to produce bits with entropy.
The amount of information necessary to justify the expected min-entropy rate depends on the type of entropy source included in the product.
For developer provided entropy sources, in order to justify the min-entropy rate, it is expected that a large number of raw source bits will be collected, statistical tests will be performed, and the min-entropy rate determined from the statistical tests. While no particular statistical tests are required at this time, it is expected that some testing is necessary in order to determine the amount of min-entropy in each output.
For third party provided entropy sources, in which the TOE vendor has limited access to the design and raw entropy data of the source, the documentation will indicate an estimate of the amount of min-entropy obtained from this third-party source. It is acceptable for the vendor to "assume" an amount of min-entropy, however, this assumption must be clearly stated in the documentation provided. In particular, the min-entropy estimate must be specified and the assumption included in the ST.
Regardless of type of entropy source, the justification will also include how the DRBG is initialized with the entropy stated in the ST, for example by verifying that the min-entropy rate is multiplied by the amount of source data used to seed the DRBG or that the rate of entropy expected based on the amount of source data is explicitly stated and compared to the statistical rate. If the amount of source data used to seed the DRBG is not clear or the calculated rate is not explicitly related to the seed, the documentation will not be considered complete.
The entropy justification shall not include any data added from any third-party application or from any state saving between restarts.

~~D~~

C.3 Operating Conditions

The entropy rate may be affected by conditions outside the control of the entropy source itself. For example, voltage, frequency, temperature, and elapsed time after power-on are just a few of the factors that may affect the operation of the entropy source. As such, documentation will also include the range of operating conditions under which the entropy source is expected to generate random data. It will clearly describe the measures that have been taken in the system design to ensure the entropy source continues to operate under those conditions. Similarly, documentation shall describe the conditions under which the entropy source is known to malfunction or become inconsistent. Methods used to detect failure or degradation of the source shall be included.

~~D~~

C.4 Health Testing

More specifically, all entropy source health tests and their rationale will be documented. This will include a description of the health tests, the rate and conditions under which each health test is performed (e.g., at startup, continuously, or on-demand), the expected results for each health test, and rationale indicating why each test is believed to be appropriate for detecting one or more failures in the entropy source.

~~E.~~

Appendix D - Application Software Equivalency Guidelines

D.1 Introduction

The purpose of equivalence in PP-based evaluations is to find a balance between evaluation rigor and commercial practicability—to ensure that evaluations meet customer expectations while recognizing that there is little to be gained from requiring that every variation in a product or platform be fully tested. If a product is found to be compliant with a PP on one platform, then all equivalent products on equivalent platforms are also considered to be compliant with the PP.

A Vendor can make a claim of equivalence if the Vendor believes that a particular instance of their Product implements PP-specified security functionality in a way equivalent to the implementation of the same functionality on another instance of their Product on which the functionality was tested. The Product instances can differ in version number or feature level (model), or the instances may run on different platforms. Equivalency can be used to reduce the testing required across claimed evaluated configurations. It can also be used during Assurance Maintenance to reduce testing needed to add more evaluated configurations to a certification.

These equivalency guidelines do not replace Assurance Maintenance requirements or NIAP Policy #5 requirements for CAVP certificates. Nor may equivalency be used to leverage evaluations with expired certifications.

These Equivalency Guidelines represent a shift from complete testing of all product instances to more of a risk-based approach. Rather than require that every combination of product and platform be tested, these guidelines support an approach that recognizes that products are being used in a variety of environments—and often in cloud environments over where the vendor (and sometimes the customer) have little or no control over the underlying hardware. Developers should be responsible for the security functionality of their applications on the platforms they are developed for—whether that is an operating system, a virtual machine, or a software-based execution environment such as a container. But those platforms may themselves run within other environments—virtual machines or operating systems—that completely abstract away the underlying hardware from the application. The developer should not be held accountable for security functionality that is implemented by platform layers that are abstracted away. The implication is that not all security functionality will necessarily be tested for all platform layers down to the hardware for all evaluated configurations—especially for applications developed for software-based execution environments such as containers. For these cases, the balancing of evaluation rigor and commercial practicability tips in favor of practicability. Note that this does not affect the requirement that at least one product instance be fully tested on at least one platform with cryptography mapped to a CAVP certificate.

Equivalency has two aspects:

1. ***Product Equivalence:*** Products may be considered equivalent if there are no differences between Product Models and Product Versions with respect to PP-specified security functionality.
2. ***Platform Equivalence:*** Platforms may be considered equivalent if there are no significant differences in the services they provide to the Product—or in the way the platforms provide those services—with respect to PP-specified security functionality.

The equivalency determination is made in accordance with these guidelines by the Validator and Scheme using information provided by the Evaluator/Vendor.

D.2 Approach to Equivalency Analysis

There are two scenarios for performing equivalency analysis. One is when a product has been certified and the vendor wants to show that a later product should be considered certified due to equivalence with the earlier product. The other is when multiple product variants are going though evaluation together and the vendor would like to reduce the amount of testing that must be done. The basic rules for determining equivalence are the same in both cases. But there is one additional consideration that applies to equivalence with previously certified products. That is, the product with which equivalence is being claimed must have a valid certification in accordance with scheme rules and the Assurance Maintenance process must be followed. If a product's certification has expired, then equivalence cannot be claimed with that product.

When performing equivalency analysis, the Evaluator/Vendor should first use the factors and guidelines for Product Model equivalence to determine the set of Product Models to be evaluated. In general, Product Models that do not differ in PP-specified security functionality are considered equivalent for purposes of evaluation against the AppPP.

If multiple revision levels of Product Models are to be evaluated—or to determine whether a revision of an evaluated product needs re-evaluation—the Evaluator/Vendor and Validator should use the factors and guidelines for Product Version equivalence to analyze whether Product Versions are equivalent.

Having determined the set of Product Models and Versions to be evaluated, the next step is to determine the set of Platforms that the Products must be tested on.

Each non-equivalent Product for which compliance is claimed must be fully tested on each non-equivalent platform for which compliance is claimed. For non-equivalent Products on equivalent platforms, only the differences that affect PP-specified security functionality must be tested for each product.

"Differences in PP-Specified Security Functionality" Defined

If PP-specified security functionality is implemented by the TOE, then differences in the actual implementation between versions or product models break equivalence for that feature. Likewise, if the TOE implements the functionality in one version or model and the functionality is implemented by the platform in another version or model, then equivalence is broken. If the functionality is implemented by the platform in multiple models or versions on equivalent platforms, then the functionality is considered different if the product invokes the platform differently to perform the function.

D.3 Specific Guidance for Determining Product Model Equivalence

Product Model equivalence attempts to determine whether different feature levels of the same product across a product line are equivalent for purposes of PP testing. For example, if a product has a “basic” edition and an “enterprise” edition, is it necessary to test both models? Or does testing one model provide sufficient assurance that both models are compliant?

Product models are considered equivalent if there are no differences that affect PP-specified security functionality—as indicated in Table 1.

Factor	Same/Different	Guidance
PP-Specified Functionality	Same	If the differences between Models affect only non-PP-specified functionality, then the Models are equivalent.
	Different	If PP-specified security functionality is affected by the differences between Models, then the Models are not equivalent and must be tested separately. It is necessary only to test the functionality affected by the software differences. If only differences are tested, then the differences must be enumerated, and for each difference the Vendor must provide an explanation of why each difference does or does not affect PP-specified functionality. If the Product Models are separately tested fully, then there is no need to document the differences.

Table 1. Determining Product Model Equivalence

D.4 Specific Guidance for Determining Product Version Equivalence

In cases of version equivalence, differences are expressed in terms of changes implemented in revisions of an evaluated Product. In general, versions are equivalent if the changes have no effect on any security-relevant claims about the TOE or assurance evidence. Non-security-relevant changes to TOE functionality or the addition of non-security-relevant functionality does not affect equivalence.

Factor	Same/Different	Guidance
Product Models	Different	Versions of different Product Models are not equivalent unless the Models are equivalent as defined in Section 3.
PP-Specified Functionality	Same	If the differences affect only non-PP-specified functionality, then the Versions are equivalent.
	Different	If PP-specified security functionality is affected by the differences, then the Versions are not considered equivalent and must be tested separately. It is necessary only to test the functionality affected by the changes. If only the differences are tested, then for each difference the Vendor must provide an explanation of why the difference does or does not affect PP-specified functionality. If the Product Versions are separately tested fully, then there is no need to document the differences.

Table 2. Factors for Determining Product Version Equivalence

D.5 Specific Guidance for Determining Platform Equivalence

Platform equivalence is used to determine the platforms that equivalent versions of a Product must be tested on. Platform equivalence analysis done for one software application cannot be applied to another software application. Platform equivalence is not general—it is with respect to a particular application.

Product Equivalency analysis must already have been done and Products have been determined to be equivalent.

The platform can be hardware or virtual hardware, an operating system or similar entity, or a software execution environment such as a container. For purposes of determining equivalence for software applications, we address each type of platform separately. In general, platform equivalence is based on differences in the interfaces between the TOE and Platform that are relevant to the implementation of PP-specified security functionality.

D.5.1 Platform Equivalence—Hardware/Virtual Hardware Platforms

If an Application runs directly on hardware without an operating system—or directly on virtualized hardware without an operating system—then platform equivalence is based on processor architecture and instruction sets. In the case of virtualized hardware, it is the virtualized processor and architecture that are presented to the application that matters—not the physical hardware.

Platforms with different processor architectures and instruction sets are not equivalent. This is not likely to be an issue for equivalency analysis for applications since there is likely to be a different version of the application for different hardware environments. Equivalency analysis becomes important when comparing processors with the same architecture. Processors with the same architecture that have instruction sets that are subsets or supersets of each other are not disqualified from being equivalent for purposes of an App evaluation. If the application takes the same code paths when executing PP-specified security functionality on different processors of the same family, then the processors can be considered equivalent with respect to that application. For example, if an application follows one code path on platforms that support the AES-NI instruction and another on platforms that do not, then those two platforms are not equivalent with respect to that application functionality. But if the application follows the same code path whether or not the platform supports AES-NI, then the platforms are equivalent with respect to that functionality.

The platforms are equivalent with respect to the application if the platforms are equivalent with respect to all PP-specified security functionality.

Factor	Same/Different/None	Guidance
Platform Architectures	Different	Platforms that present different processor architectures and instruction sets to the application are not equivalent.
PP-Specified Functionality	Same	For platforms with the same processor architecture, the platforms are equivalent with respect to the application if execution of all PP-specified security functionality follows the same code path on both platforms.

Table 3. Factors for Determining Hardware/Virtual Hardware Platform Equivalence

D.5.2 Platform Equivalence—OS Platforms

For traditional applications that are built for and run on operating systems, platform equivalence is determined by the interfaces between the application and the operating system that are relevant to PP-specified security functionality. Generally, these are the processor interface, device interfaces, and OS APIs. The following factors applied in order:

Factor	Same/Different/None	Guidance
Platform Architectures	Different	Platforms that run on different processor architectures and instruction sets are not equivalent.
Platform Vendors	Different	Platforms from different vendors are not equivalent.
Platform Versions	Different	Platforms from the same vendor with different major version numbers are not equivalent.
Platform Interfaces	Different	Platforms from the same vendor and major version are not equivalent if there are differences in device interfaces and OS APIs that are relevant to the way the platform provides PP-specified security functionality to the application.
Platform Interfaces	Same	Platforms from the same vendor and major version are equivalent if there are no differences in device interfaces and OS APIs that are relevant to the way the platform provides PP-specified security functionality to the application, or if the Platform does not provide such functionality to the application.

Table 4. Factors for Determining OS/VS Platform Equivalence

D.5.3 Software-based Execution Environment Platform Equivalence

If an Application is built for and runs in a non-OS software-based execution environment, such as a Container or Java Runtime, then the below criteria must be used to determine platform equivalence. The key point is that the underlying hardware (virtual or physical) and OS is not relevant to platform equivalence. This allows applications to be tested and run on software-based execution environments on any hardware—as in cloud deployments.

Factor	Same/Different/None	Guidance
Platform Type/Vendor	Different	Software-based execution environments that are substantially different or come from different vendors are not equivalent. For example, a java virtual machine is not the same as a container. A Docker container is not the same as a CoreOS container.
Platform Versions	Different	Execution environments that are otherwise equivalent are not equivalent if they have different major version numbers.
PP-Specified Security Functionality	Same	All other things being equal, execution environments are equivalent if there is no significant difference in the interfaces through which the environments provide PP-specified security functionality to applications.

Table 5. Factors for Software-based Execution Environment Platform Equivalence

D.6 Level of Specificity for Tested Configurations and Claimed Equivalent Configurations

In order to make equivalency determinations, the vendor and evaluator must agree on the equivalency claims. They must then provide the scheme with sufficient information about the TOE instances and platforms that were evaluated, and the TOE instances and platforms that are claimed to be equivalent.

The ST must describe all configurations evaluated down to processor manufacturer, model number, and microarchitecture version.

The information regarding claimed equivalent configurations depends on the platform that the application was developed for and runs on.

Bare-Metal Applications

For applications that run without an operating system on bare-metal or virtual bare-metal, the claimed configuration must describe the platform down to the specific processor manufacturer, model number, and microarchitecture version. The Vendor must describe the differences in the TOE with respect to PP-specified security functionality and how the TOE functions differently to leverage platform differences (e.g., instruction set extensions) in the tested configuration versus the claimed equivalent configuration.

Traditional Applications

For applications that run with an operating system as their immediate platform, the claimed configuration must describe the platform down to the specific operating system version. If the platform is a virtualization system, then the claimed configuration must describe the platform down to the specific virtualization system version. The Vendor must describe the differences in the TOE with respect to PP-specified security functionality and how the TOE functions differently to leverage platform differences in the tested configuration versus the claimed equivalent configuration. Relevant platform differences could include instruction sets, device interfaces, and OS APIs invoked by the TOE to implement PP-specified security functionality.

Software-Based Execution Environments

For applications that run in a software-based execution environment such as a Java virtual machine or a Container, then the claimed configuration must describe the platform down to the specific version of the software execution environment. The Vendor must describe the differences in the TOE with respect to PP-specified security functionality and how the TOE functions differently to leverage platform differences in the tested configuration versus the claimed equivalent configuration.

Appendix E - References

Identifier	Title

Common Criteria for Information Technology Security Evaluation -

- Part 1: Introduction and General Model, CCMB-

~~2012~~

- 2017-

~~09~~

- 04-001, Version 3.1, Revision

~~4~~

- 5,

~~September 2012~~

- April 2017.
- Part 2: Security Functional Components, CCMB-

~~2012~~

- 2017-

~~09~~

[CC]

- 04-002, Version 3.1, Revision

~~4~~

- 5,

~~September 2012~~

- April 2017.
- Part 3: Security Assurance Components, CCMB-

~~2012~~

- 2017-

~~09~~

- 04-003, Version 3.1, Revision

~~4~~

- 5,

~~September 2012~~

- April 2017.

[CEM] Common Evaluation Methodology for Information Technology Security - Evaluation Methodology, CCMB-~~2012~~2017-~~09~~04-004, Version 3.1, Revision ~~4~~5, ~~September 2012~~April 2017.

~~[CESG]~~ ~~CESG - End User Devices Security and Configuration Guidance~~

~~[CSA]~~ ~~Computer Security Act of 1987, H.R. 145, June 11, 1987.~~

~~[~~[OMB] Reporting Incidents Involving Personally Identifiable Information and Incorporating the Cost for Security in Agency Information Technology Investments, OMB M-06-19, July 12, 2006.

Appendix F

~~-~~

- Acronyms

Acronym	Meaning	
ADB	Android Debug Bridge	
AES	Advanced Encryption Standard	
ANSI	American National Standards Institute	
API	Application Programming Interface	
API	Application Programming Interface	
APK	Android Application Package	
APPX	Windows Universal Application Package	
~~API~~	~~Application Programming Interface~~ ASLR	Address Space Layout Randomization
ASLR	Address Space Layout Randomization	
~~BAR~~	~~Blackberry Application Package~~	
BIOS	Basic Input/Output System	
~~CDSA~~ CC	Common ~~Data Security Architecture~~Criteria	
~~CESG~~	~~Communications-Electronics Security Group~~ CEM	Common Evaluation Methodology
CMC	Certificate Management over CMS	
CMS	Cryptographic Message Syntax	
CN	Common Names	
CRL	Certificate Revocation List	
CSA	Computer Security Act	

DEP	Data Execution Prevention
DES	Data Encryption Standard
DHE	Diffie-Hellman Ephemeral
DMG	Apple Disk Image
DNS	Domain Name System
DPAPI	Data Protection Application Programming Interface
DRBG	Deterministic Random Bit Generator
DSS	Digital Signature Standard
DSS	Digital Signature Standard
DT	Date/Time Vector
DTLS	Datagram Transport Layer Security
EAP	Extensible Authentication Protocol
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
ELF	Executable and Linkable Format
EMET	Enhanced Mitigation Experience Toolkit
EP	Extended Package
EST	Enrollment over Secure Transport
FIPS	Federal Information Processing Standards
~~DSS~~	~~Digital Signature Standard~~
GPS	Global Positioning System
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
~~DSS~~	~~Digital Signature Standard~~
IANA	Internet Assigned Number Authority
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPA	iOS Package archive
IR	Intermediate Integer
ISO	International Organization for Standardization
IT	Information Technology
ITSEF	Information Technology Security Evaluation Facility
JNI	Java Native Interface
LDAP	Lightweight Directory Access Protocol
MIME	Multi-purpose Internet Mail Extensions
MPKG	Meta Package
MSI	Microsoft Installer
NFC	Near Field Communication
NIAP	National Information Assurance Partnership
NIST	National Institute of Standards and Technology
OCSP	Online Certificate Status Protocol
OID	Object Identifier
OMB	Office of Management and Budget
OS	Operating System
PDF	Portable Document Format
PE	Portable Executable
PID	Process Identifier
PII	Personally Identifiable Information
PKG	Package file
PKI	Public Key Infrastructure
PP	Protection Profile
~~IT~~ PP ~~Information Technology~~	Protection Profile
RBG	Random Bit Generator
RFC	Request for Comment
RNG	Random Number Generator
RNGVS	Random Number Generator Validation System
S/MIME	Secure/Multi-purpose Internet Mail Extensions

SAN	Subject Alternative Name
SAR	Security Assurance Requirement
SAR	Security Assurance Requirement
SE	Security Enhancements
SFR	Security Functional Requirement
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
~~S/MIME~~	~~Secure/Multi-purpose Internet Mail Extensions~~
SIP	Session Initiation Protocol
SP	Special Publication
SSH	Secure Shell
ST	Security Target
SWID	Software Identification
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSS	TOE Summary Specification
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
XCCDF	eXtensible Configuration Checklist Description Format
XOR	Exclusive Or
app	Application