Protection Profile for Email Clients

1 April 2014

Version 1.0

Revision History

Version	Date	Description
1.0	1 April 2014	Email Client PP

1 Introduction

This document provides a baseline set of Security Functional Requirements (SFRs) for an Email client, which is the Target of Evaluation (TOE).

Email clients are applications used to send, receive, access and manage email provided by an email server. The complexity of email content and email clients has grown over time. Modern email clients can render HTML as well as plaintext, and may include functionality to display common attachment formats, such as Adobe PDF and Microsoft Word documents. Some email clients allow their functionality to be modified by users through the addition of extensions or plug-ins. Protocols have also been defined for communicating between email clients and servers. Some clients support multiple protocols for doing the same task, allowing them to be configured according to email server specifications.

The complexity and rich feature set of modern email clients make them a target for attackers, introducing security concerns. This document is intended to facilitate the improvement of email client security by requiring use of operating system security services, cryptographic standards, and environmental mitigations. Additionally, the requirements in this document define acceptable behavior for email clients regardless of the security features provided by the operating system.

The requirements apply to all email clients that run on any operating system, regardless of the composition of the underlying platform. For purposes of this document, an *application* is defined as software that runs on an operating system and performs tasks on behalf of the user or owner of the platform. An *email client* is an application that retrieves and manages email provided by an email server. *Extensions and plug-ins* are code packages that can be loaded by the email client to introduce new or specialized functionality to the client.

1.1 Overview of the TOE

The Target of Evaluation (TOE) in this document is an email client application running on a desktop or mobile operating system.

1.2 Usage of the TOE

Email clients are used to retrieve email from a Mail User Agent (MUA) or to send mail via a Mail Transfer Agent (MTA). MUA and MTA functionality may be available in the same product, and the term mail server may apply to either an MUA or MTA. Access to resources via an email client can be done over the Internet, or within a closed network (Intranet). Although some email clients integrate the capability to manage calendars, contacts, etc., such functionality is out of scope for this protection profile.

[USE CASE] Sending, receiving, accessing, managing and displaying email

Email clients are used for sending, receiving, viewing, accessing, managing email in coordination with a mail server. Email clients can render HTML as well as plaintext, and can display common attachment formats.

2 SECURITY PROBLEM DESCRIPTION

The following describes the problems that compliant TOEs will address.

2.1 Threats

2.1.1 Malicious or Flawed Updates

Since the most common attack vector used involves attacking unpatched versions of software containing well-known flaws, updating the email client is necessary to ensure that changes to threat environment are addressed. Timely application of patches ensures that the client is a “hard target”, thus increasing the likelihood that product will be able to maintain and enforce its security policy. However, the updates to be applied to the product must be trustable in some manner; otherwise, an attacker can write their own “update” that instead contains malicious code of their choosing, such as a rootkit, bot, or other malware. Once this “update” is installed, the attacker then has control of the system and all of its data.

[T.UNAUTHORIZED_UPDATE]

2.1.2 Malicious or Flawed Plug-ins or Extensions

Email client functionality can be extended with integration of third-party utilities and tools. This expanded set of capabilities is made possible via the use of plug-ins and extensions. The tight integration between the basic email client code and the new capabilities that plug-ins and extensions provide increases the risk that malefactors could inject serious flaws into the email client application, either maliciously by an attacker, or accidentally by a developer. These flaws enable undesirable behaviors including, but not limited to, allowing unauthorized access to sensitive information in the email client, unauthorized access to the device’s file system, or even privilege escalation that enables unauthorized access to other applications or the operating system.

[T.UNAUTHORIZED_ADDON]

2.1.3 Network Eavesdropping

Network eavesdropping involves an attacker positioning himself on the network in order to monitor transmissions between a system and the intended destination of some potentially sensitive data. With respect to email clients, this entails monitoring the transactions between the email client and the MUA and MTA.

[T.NETWORK_EAVESDROP]

2.1.4 Network Attack

Network attack is similar to network eavesdropping in that it entails an attacker positioning herself on the network. It differs from network eavesdropping in that it involves the attacker engaging in communications with a system, or modifying data between a system and some data’s legitimate destination. With respect to email clients, network attack might involve sending malicious data to the email client in order to exploit vulnerabilities that may influence its behavior, or modifying account information en route to a MUA or MTA. Email client attacks generally occur on Internet connected email client, but are not unknown to occur within closed networks. One class of attack that may exploit vulnerabilities in the email client software or email client extensions involves the use of phishing or other social engineering technique to deliver the exploit. In this kind of attack, a user opens a malicious attachment or clicks on a link to a malicious website, which then executes the exploit that compromises the email client, often with no indication to the user.

[T.NETWORK_ATTACK]

2.2 **Assumptions**

The basic assumptions relevant to the functional and security capabilities of the underlying platform and the environment in which the TOE is expected to operate are defined in Annex A.

3 Security Objectives

Compliant TOEs will provide security functionality to address security objectives as enumerated below, and to implement policies that address additional threats to the TOE. The following sections provide a description of this functionality, given the threats enumerated above.

3.1 Security Objectives of the TOE

3.1.1 Protected Communications

To address the network attack and network eavesdropping threats described in the Threats section, the TOE must provide for protected communications between the email client and a given email server in instances where such protected communications are desirable. The data between these two entities in the operational environment are protected via a trusted path, implemented using one or more of these standard protocols: Transport Layer Security (TLS), Secure Multipurpose Internet Mail Extensions (S/MIME) and Simple Authentication and Security Layer (SASL).

[O.COMMS]

3.1.2 TOE Configuration

In order to protect sensitive data stored (either temporarily or permanently) or processed by the email client, conformant TOEs will provide the capability to define, configure, and apply security policies defined by the administrator. If enterprise policies are configured by the administrator for the TOE, these must take precedence over any user-defined settings.

[O.CONFIG]

3.1.3 Integrity of TOE

To ensure the integrity of the email client is maintained, conformant TOEs will perform self-tests to insure the integrity of software and data has been maintained.

To address issues associated with malicious or flawed email client software, plug-ins or extensions, conformant TOEs must implement mechanisms to ensure the integrity of email client software, plug-ins and extensions, and to ensure that they come from legitimate sources. The TOE must provide mechanisms and enforce policies that enable any client software, plug-ins and extensions, as well as any subsequent updates to them, to be verified upon installation and execution. In addition, the TOE shall also control the downloading and launching of executables.

[O.INTEGRITY]

3.1.4 Secure Storage of Sensitive Information

Email clients handle many types of potentially sensitive user information (e.g., passwords, cryptographic keys, digital certificates). It is critical that clients protect this information when it is stored. The email client shall make use of platform encryption and authentication mechanisms and libraries to protect this information rather than mechanisms and libraries that are part of the email client itself.

[O.STORAGE]

4 Security Requirements

Some of 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*, with additional extended functional components. This section addresses Security Functional Requirements that may be met by either the TOE itself or by the TOE platform. In the case of email clients that are designed to run on mobile platforms, the TOE shall use the mobile platform's data-at-rest (DAR) protections.

4.1 Conventions

The CC defines operations on Security Functional Requirements: assignments, selections, assignments within selections and refinements. This document uses the following font conventions to identify the operations defined by the CC:

- Assignment: Indicated with *italicized* text;
- Refinement made by PP author: Indicated by the word "Refinement" in **bold text** after the element number with additional text in **bold** text and deletions with strikethroughs, if necessary;
- Selection: Indicated with underlined text;
- Assignment within a Selection: Indicated with *italicized and underlined* text;
- Iteration: Indicated by appending the iteration number in parenthesis, e.g., (1), (2), (3).

Explicitly stated SFRs are identified by having a label 'EXT' after the requirement name for TOE SFRs.

4.2 Security Functional Requirements

This section addresses those Security Functional Requirements that may be met by the TOE.

4.2.1 Class: Cryptographic Support (FCS)

4.2.1.1 Cryptographic Key Management

FCS_CKM_EXT.1 Cryptographic Key Storage

FCS_CKM_EXT.1.1 The TSF shall store persistent secrets and private keys when not in use in platform-provided key storage.

Application Note:

This requirement ensures that persistent secrets (e.g. passwords, certificates, other credentials, secret keys) and private keys are stored securely when not in use.

This requirement mandates persistent secrets and private keys used by the Email Client will be stored by the platform.

Assurance Activity:

The evaluator will check the TSS to ensure that it lists each persistent secret (password, credential, or secret key) and private key needed to meet the requirements in the ST. For each of these items, the evaluator will confirm that the TSS lists how the item is identified, for what purpose it is used, and how it is stored. The evaluator then performs the following actions.

Persistent secrets and private keys manipulated by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the persistent secrets and private keys listed as being stored by the platform in the Email Client ST are identified as being protected in that platform's ST.

Persistent secrets and private keys manipulated by the TOE

The evaluator reviews the TSS for to determine that it makes a case that, for each item listed as being manipulated by the TOE, it is not written unencrypted to persistent memory, and that the item is stored by the platform.

4.2.2 Class: User Data Protection (FDP)

4.2.2.1 S/MIME for Data Protection

FDP_SMIME_EXT.1.1 The TSF shall use Secure/Multipurpose Internet Mail Extensions (S/MIME) to sign, verify, encrypt, and decrypt mail.

Application Note: *S/MIME is used to sign messages at the request of the user (FMT_SMF.1.1 Function 7) upon sending email via the Mail Transfer Agent. S/MIME is used by a recipient to verify the digital signature on a signed message upon receipt or viewing of the message. Encryption of messages occurs at the request of the user (FMT_SMF.1.1 Function 8) upon sending email via the Mail Transfer Agent, and decryption of emails occurs upon receipt or viewing of the message. Note that this requirement does not mandate that S/MIME be used for all incoming/outgoing messages, or that the TOE automatically encrypt and/or sign/verify all sent or received messages. This requirement only specifies that the mechanism for digital signature and encryption must be S/MIME.*

Assurance Activity:

The evaluator shall verify that the TSS contains a description of the S/MIME implementation and its use to protect mail from undetected modification using digital signatures and unauthorized disclosure using encryption. The evaluator shall verify that the TSS describes whether signature verification and decryption occur at receipt or viewing of the message contents, and whether messages are stored with their S/MIME envelopes.

The evaluator shall ensure that the AGD guidance includes instructions for configuring a certificate for S/MIME use and instructions for signing and encrypting email.

Tests for this element are performed in conjunction with tests for FCS_SMIME_EXT.1, FDP_NOT_EXT.1, and FMT_SMF.1.

4.2.2.2 S/MIME for Access to Certificates

FDP_SMIME_EXT.1.2 The [selection: TSF, TOE platform] shall provide access to a certificate repository for the purposes of S/MIME encryption.

Application Note: *Encryption requires access to the recipient's public key in the form of an X.509v3 certificate; thus, these public keys must be accessible via a local or remote repository. Configuration of this repository is required to be performed either by the TOE or TOE platform according to FMT_SMF.1.1 Function 10.*

Assurance Activity:

The evaluator shall verify that the TSS contains a description of the certificate repository implementation which must at least indicate whether the repository is local, remote, or both, and whether the repository is managed by the TOE or by the platform. If the description indicates that a remote repository may be used, the evaluator shall ensure that the description indicates which protocols may be used for interfacing with that repository. The evaluator shall also verify that the description of the local repository indicates whether certificates are loaded automatically or must be added manually. If loaded automatically, the evaluator shall ensure that a description of the automatic mechanism is provided (for example, when signed emails are received the associated certificates are loaded into the local repository).

The evaluator shall also verify that the AGD guidance contains instructions for configuring the certificate repository used for S/MIME encryption. If the TSS indicates that the repository is local, the evaluator shall verify that the instructions indicate how certificates are loaded and removed. If the TSS indicates that the repository is remote, the evaluator shall verify that the instruction indicate how a user or administrator configures the remote server information.

Tests for this element are performed in conjunction with tests for FCS_SMIME_EXT.1 and FMT_SMF.1.

4.2.2.3 Data Notification

FDP_NOT_EXT.1 Notification of Email Receipt

FDP_NOT_EXT.1.1 The TSF shall display notifications of email receipt.

Application Note: *The notification may be visual or auditory and may or may not additional information about the email(s) or the number of emails received. Some notification implementations pop-up briefly and display the subject, sender, and partial content of the email. FMT_SMF.1.1 Function 1 requires the ability to disable these notifications. If the notification contains any email content, FMT_SMF.1.1 Function 9, which requires the ability to disable the display content in the notification, must be included in the ST.*

Assurance Activity:

The evaluator shall ensure that the TSS describes email notifications, including whether the notification is visual or auditory, whether it includes any information about the email(s), and, if it contains information about the email(s), what information (such as number, subject, sender, or partial contents). If the TSS indicates that any email content is included in the notification, the evaluator shall ensure that the ST includes function 9 in FMT_SMF.1.1.

The evaluator shall verify that the AGD guidance provides a description (with any appropriate visual figures) of the notification(s) for users. The AGD guidance must also describe how users and/or administrators may disable notifications. If the TSS indicates that notifications contain mail contents, the evaluator shall ensure that the AGD guidance provides instructions for disabling the display of content in notifications.

The evaluator shall perform the following test:

Test 1: The evaluator shall send an email to the client and verify that the notification(s) appear(s) as described.

FDP_NOT_EXT.2 Notification of S/MIME Status

FDP_NOT_EXT.2.1 The TSF shall display a notification of the S/MIME status of received emails upon viewing.

Application Note: *S/MIME status is whether the email has been signed or encrypted and whether the signature verifies and the associated certificate validates. This notification satisfies FIA_X509_EXT.2.6. This notification must at least display when the email content is viewed. Many implementations also display the S/MIME status of each email when all emails are viewed as a list.*

Assurance Activity:

The evaluator shall ensure that the TSS describes notifications of S/MIME status, including whether S/MIME status is also indicated upon viewing a list of emails.

The evaluator shall verify that the AGD guidance proves a description (with appropriate visual figures) of the S/MIME status notification(s), including how each of encryption, verified and validated signature, and unverified and unvalidated signature are indicated.

The evaluator shall perform the following tests and may perform them in conjunction with the tests for FCS_SMIME_EXT.1:

Test 1: The evaluator shall send the client an unencrypted and unsigned email and verify that no notifications are present upon viewing.

Test 2: The evaluator shall send the client an encrypted email and verify that the encrypted notification is present upon viewing.

Test 3: The evaluator shall send the client a valid signed email and verify that the signed notification is present upon viewing.

Test 4: The evaluator shall send the client an invalid signed email (for example, using a certificate that does not contain the correct email address or a certificate that does not chain to the root store) and verify that the invalid signature notification is present upon viewing.

FDP_NOT_EXT.3 Notification of URI

FDP_NOT.EXT.3.1 The TSF shall display the full Uniform Resource Identifier (URI) of any embedded links.

Application Note: *Embedded links are HTML URI objects which may have a tag (such as a word, phrase, icon, or picture) that obfuscates the URI of the link. The intent of this requirement is de-obfuscate the link. The URI may be displayed as a "mouse-over" event or may be rendered next to the tag.*

Assurance Activity:

The evaluator shall verify that the TSS includes a description of how embedded links are rendered and the method by which the URI of the link is displayed.

The evaluator shall ensure that the AGD guidance includes instruction (with any appropriate visual figures) for viewing the URI of an embedded link.

The evaluator shall perform the following test:

Test 1: The evaluator shall send the client an HTML message with an embedded link whose tag is not the URI itself (for example, "click here"). The evaluator shall view the message and *following the instructions in the AGD guidance verify that the full URI of the embedded link is displayed.*

4.2.2.4 Rendering of Message Content

FDP_REN_EXT.1.1 The TSF shall have a plaintext only mode.

Application Note: *Plaintext only mode prevents the automatic downloading of images and rendering and execution of embedded objects such as HTML or JavaScript objects. FMT_SMF.1.1 function 3 addresses configuration of this mode.*

Assurance Activity:

The evaluator shall ensure that the TSS describes plaintext only mode for sending and receiving messages. The evaluator shall verify that the TSS describes whether the TOE is capable of rendering and executing HTML or JavaScript. If the TOE can render or execute HTML or JavaScript, this description must indicate how the TOE handles received messages that contain HTML or JavaScript while in plaintext only mode, and the evaluator shall ensure that that description indicates that embedded objects of these types are not rendered or executed and that images are not automatically downloaded.

The evaluator shall examine the AGD guidance and verify that it contains instructions for enabling plaintext only mode.

The evaluator shall perform the following tests:

Test 1: (conditional) If the TOE is capable of rendering HTML, the evaluator shall send a message to the client containing HTML embedded objects and shall verify that the HTML renders. The evaluator shall then enable plaintext only mode and verify that the HTML does not render.

Test 2: (conditional) If the TOE is capable of rendering and executing JavaScript, the evaluator shall send a message to the client containing JavaScript embedded objects and shall verify that the JavaScript renders and executes. The evaluator shall then enable plaintext only mode and verify that the JavaScript does not render or execute.

4.3 **TOE or TOE Platform Security Functional requirements**

This section addresses Security Functional Requirements that may be met by either the TOE itself, by the TOE platform, or by a combination of the TOE and the TOE platform.

4.3.1 Class: Cryptographic Support (FCS)

4.3.1.1 Cryptographic Key Management

FCS_CKM.1 Cryptographic Key Generation

FCS_CKM.1.1(1) **Refinement:** The [selection: TSF, TOE platform] shall generate **asymmetric** cryptographic keys **used for key establishment** in accordance with

- NIST Special Publication 800-56A, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography” for elliptic curve-based key establishment schemes and implementing “NIST curves” P-256, P-384 and [selection: P-521, no other curves] (as defined in FIPS PUB 186-4, “Digital Signature Standard”)

- NIST Special Publication 800-56B, “Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography” for RSA-based key establishment schemes

[selection:

- *NIST Special Publication 800-56A, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography” for finite field-based key establishment schemes;*
- *No other schemes*]

and specified cryptographic key sizes *equivalent to, or greater than, a symmetric key strength of 112 bits*.

Application Note:

This component requires that the TSF or the TOE platform be able to generate the public/private key pairs that are used for key establishment purposes for the various cryptographic protocols used by the TOE (e.g., trusted channel). If multiple schemes are supported, then the ST author should iterate this requirement to capture this capability. The scheme used will be chosen by the ST author from the selection.

Since the domain parameters to be used are specified by the requirements of the protocol in this PP, it is not expected that the TOE will generate domain parameters, and therefore there is no additional domain parameter validation needed when the TOE complies with the protocols specified in this PP.

The generated key strength of 2048-bit DSA and RSA keys need to be equivalent to, or greater than, a symmetric key strength of 112 bits. See NIST Special Publication 800-57, “Recommendation for Key Management” for information about equivalent key strengths.

RSA and elliptic curve-based schemes are required in order to comply with the required ciphersuites in FCS_TLSC_EXT.1.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the key establishment claimed in that platform's ST contains the key establishment requirement in the Email Client’s ST. The evaluator shall also examine the TSS of the Email Client’s ST to verify that it describes (for each supported platform) how the key establishment functionality is invoked (it should be noted that this may be through a mechanism that is not implemented by the Email Client; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the TOE

This assurance activity will verify the key generation and key establishments schemes used on the TOE.

Key Generation:

The evaluator shall verify the implementation of the key generation routines of the supported schemes using the applicable tests below.

Key Generation for RSA-Based Key Establishment Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE 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:

- *Random Primes:*
    - *Provable primes*
    - *Probable primes*
- *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 the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

Key Generation for Finite-Field Cryptography (FFC) – Based 56A Schemes

<u>FFC Domain Parameter and Key Generation Tests</u>

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 <= q-1*
- *len(q) + 64 bit output of RBG, followed by a mod q-1 operation where 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 mod p = 1*
- *g^x mod p = y*

for each FFC parameter set and key pair.

Key Generation for Elliptic Curve Cryptography (ECC)- Based 56A Schemes

ECC Key Generation Test

For each supported NIST curve, i.e., P-256, P-284 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.

ECC Public Key Verification (PKV) Test

For each supported NIST curve, i.e., P-256, P-284 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 Establishment Schemes

The evaluator shall verify the implementation of the key establishment schemes of the 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.*

<u>Function Test</u>

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 public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information field OI and TOE id fields.

If the TOE does not use a KDF defined in SP 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 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.

<u>Validity Test</u>

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 public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE 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 other information field OI, 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 static private key to assure the TOE 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 results with the results using a known good implementation verifying that the TOE detects these errors.

SP800-56B Key Establishment Schemes

At this time, detailed test procedures for RSA-based key establishment schemes are not available. In order to show that the TSF complies with 800-56B, depending on the selections made, the evaluator shall ensure that the TSS contains the following information:

- *The TSS shall list all sections of the appropriate 800-56B standard(s) 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 800-56B (as selected), any omission of functionality related to "shall" or "should" statements shall be described.

FCS_CKM_EXT.4 Cryptographic Key Destruction

FCS_CKM_EXT.4.1 The [selection: TSF, TOE platform] shall zeroize all plaintext secret and private cryptographic keys and CSPs when no longer required.

Application Note:

The ST author should select the platform if the Email Client performs no operations using plaintext secret, private cryptographic keys, and CSPs.

Any security related information (such as keys, authentication data, and passwords) must be zeroized when no longer in use to prevent the disclosure or modification of security critical data.

The zeroization indicated above applies to each intermediate storage area for plaintext key and Cryptographic Service Provider (CSP) (i.e., any storage, such as memory buffers, that is included in the path of such data) upon the transfer of the key/CSP to another location.

Since the TOE does not include the host IT environment, the extent of this capability is necessarily somewhat limited. For the purposes of this requirement, it is sufficient for the TOE to invoke the correct underlying functions of the host to perform the zeroization--it does not imply that the TOE has to include a kernel-mode memory driver to ensure the data are zeroized. It is assumed that the host platform appropriately performs zeroization of key material in its internal processes.

Assurance Activity:

Requirement met by the platform

The evaluator shall check to ensure the TSS describes each of the secret keys (keys used for symmetric encryption), private keys, and CSPs used to generate key that are not otherwise covered by the FCS_CKM_EXT.4 requirement levied on the TOE.

For each platform listed in the ST, the evaluator shall examine the TSS of the ST of the platform to ensure that each of the secret keys, private keys, and CSPs used to generate key listed above are covered.

Requirement met by Email Client

The evaluator shall check to ensure the TSS describes each of the secret keys (keys used for symmetric encryption), private keys, and CSPs used to generate key; when they are zeroized (for example, immediately after use, on system shutdown, etc.); and the type of zeroization procedure that is performed (overwrite with zeros, overwrite three times with random pattern, etc.). If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the zeroization procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are zeroized by overwriting once with zeros, while secret keys stored on the internal hard drive are zeroized by overwriting three times with a random pattern that is changed before each write"). If a read-back is done to verify the zeroization, this shall be described as well.

For each key clearing situation described in the TSS the evaluator shall repeat the following test.

Test 1: The evaluator shall utilize appropriate combinations of specialized operational environment and development tools (debuggers, simulators, etc.) for the TOE and instrumented TOE builds to test that keys are cleared correctly, including all intermediate copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.

Cryptographic TOE implementations in software shall be loaded and exercised under a debugger to perform such tests. The evaluator shall perform the following test for each key subject to clearing, including intermediate copies of keys that are persisted encrypted by the TOE:

- *Load the instrumented TOE build in a debugger.*
- *Record the value of the key in the TOE subject to clearing.*
- *Cause the TOE to perform a normal cryptographic processing with the key from #1.*
- *Cause the TOE to clear the key.*
- *Cause the TOE to stop the execution but not exit.*
- *Cause the TOE to dump the entire memory footprint of the TOE into a binary file.*
- *Search the content of the binary file created in #4 for instances of the known key value from #1.*

The test succeeds if no copies of the key from #1 are found in step #7 above and fails otherwise.

The evaluator shall perform this test on all keys, including those persisted in encrypted form, to ensure intermediate copies are cleared.

Test 2: In cases where the TOE is implemented in firmware and operates in a limited operating environment that does not allow the use of debuggers, the evaluator shall utilize a simulator for the TOE on a general purpose operating system. The evaluator shall provide a rationale explaining the instrumentation of the simulated test environment and justifying the obtained test results.

4.3.1.2 Cryptographic Operation

FCS_COP.1(1) Cryptographic operation (Encryption and Decryption)

FCS_COP.1.1(1) The [selection: TSF, TOE platform] shall perform [***encryption/decryption***] in accordance with a specified cryptographic algorithm

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

[selection:

- *AES-GCM (as defined in NIST SP 800-38D),*
- *No other mode*]

and cryptographic key sizes 128-bit, 256-bit.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the encryption/decryption function(s) claimed in that platform's ST contains the encryption/decryption function(s) in the Email Client's ST. The evaluator shall also examine the TSS of the Email Client's ST to verify that it describes (for each supported platform) how the encryption/decryption functionality is invoked for each mode and key size selected in the Email Client's ST (it should be noted that this may be through a mechanism that is not implemented by the Email Client; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the Email Client

AES-CBC Tests

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 of AES-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 of AES-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 of AES-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 Test

The evaluator shall test the authenticated encrypt functionality of AES-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.

FCS_COP.1(2) Cryptographic operation (Hashing)

FCS_COP.1.1(2) The [selection: TSF, TOE platform] shall perform cryptographic hashing in accordance with a specified cryptographic algorithm SHA-1, SHA-256, SHA-384, and [selection: SHA-512, no other algorithms] and message digest sizes 160, 256 384, [selection: 512, no other sizes] bits that meet the following: *FIPS Pub 180-4*.

Application Note:

In future versions of this document, SHA-1 may be removed as an option. 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 and SHA-256 are required in order to comply with the required ciphersuites in FCS_TLS_EXT.1

The intent of this requirement is to specify the hashing function used for digital signature generation and verification associated with trusted updates and 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).

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the hash function(s) claimed in that platform's ST contains the hash function(s) in the Email Client's ST. The evaluator shall also examine the TSS of the Email Client's ST to verify that it describes (for each supported platform) how the hash functionality is invoked for each digest size selected in the Email Client's ST (it should be noted that this may be through a mechanism that is not implemented by the Email Client; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the Email Client

The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required hash sizes is present. The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

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 only hashes 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.

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.

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.

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.*

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.*

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 (Digital Signatures)

FCS_COP.1.1(3) The [selection: <u>TSF, TOE platform</u>] shall perform *cryptographic signature services* in accordance with the following specified cryptographic algorithms

- RSA Digital Signature Algorithm (RSA) with a key size (modulus) of 2048 bits or greater that meets FIPS PUB 186-2 or FIPS PUB 186-4, “Digital Signature Standard”,
- Elliptic Curve Digital Signature Algorithm (ECDSA) with a key size of 256 bits or greater] that meets FIPS PUB 186-4, “Digital Signature Standard” with “NIST curves” P-256, P-384 and [selection: <u>P-521, no other curves</u>] (as defined in FIPS PUB 186-4, “Digital Signature Standard”),

[selection:

- <u>Digital Signature Algorithm (DSA) with a key size (modulus) of 2048 bits or greater that meets FIPS PUB 186-4, “Digital Signature Standard”, no other cryptographic signature service</u>].

Application Note:

The Email Client must perform RSA and ECDSA digital signatures in accordance with FCS_TLS_EXT.1. The Email Client may also verify signatures on plug-ins and extensions.

If multiple schemes are supported, then the ST author should iterate this requirement to capture this capability. The scheme used will be chosen by the ST author from the selection.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the digital signature functions claimed in that platform's ST contains the digital signature functions in the Email Client's ST. The evaluator shall also examine the TSS of the Email Client’s ST to verify that it describes (for each supported platform) how the digital signature functionality is invoked for each operation they are used for in the Email Client (it should be noted that this may be through a mechanism that is not implemented by the Email Client; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the Email Client

Key Generation:

Key Generation for RSA Signature Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE 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:

- *Random Primes:*
  - *Provable primes*
  - *Probable primes*
- *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 the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

ECDSA Key Generation Tests

<u>FIPS 186-4 ECDSA Key Generation Test</u>

For each supported NIST curve, i.e., P-256, P-284 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.

<u>FIPS 186-4 Public Key Verification (PKV) Test</u>

For each supported NIST curve, i.e., P-256, P-284 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.

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

For each supported NIST curve (i.e., P-256, P-284 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.

ECDSA FIPS 186-4 Signature Verification Test

For each supported NIST curve (i.e., P-256, P-284 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

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.

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.

The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

FCS_COP.1(4) Cryptographic operation (Keyed-Hash Message Authentication)

FCS_COP.1.1(4) The [selection: TSF, TOE platform] 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], 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."

Application Note:

The intent of this requirement is to specify the keyed-hash message authentication function used when used for key establishment purposes for the various cryptographic protocols used by the TOE (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 ciphersuites in FCS_TLS_EXT.1.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the keyed-hash function(s) claimed in that platform's ST contains the keyed-hash function(s) in the Email Client's ST. The evaluator shall also examine the TSS of the Email Client's ST to verify that it describes (for each supported platform) how the keyed-hash *functionality is invoked for each mode and key size selected in the Email Client's ST (it should be noted that this may be through a mechanism that is not implemented by the Email Client; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).*

Requirement met by the Email Client

The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

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.

4.3.1.3 Random Bit Generation

FCS_RBG_EXT.1 Extended: Random Bit Generation

FCS_RBG_EXT.1.1 The [selection: TSF, TOE platform] shall perform all deterministic random bit generation services in accordance with [selection, *choose one of: NIST Special Publication 800-90A using* [selection: *Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES), Dual_EC_DRBG (any)]; FIPS Pub 140-2 Annex C: X9.31 Annex 2.4 using AES*].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [selection: *a software-based noise source, a platform-based RBG*] 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.

Application Note:

For the first selection in FCS_RBG_EXT.1.1, the ST author should select whether the TOE or the platform on which the TOE is installed provides the RBG services.

NIST Special Pub 800-90B, Annex C describes the minimum entropy measurement that will probably be required future versions of FIPS-140. If possible this should be used immediately and will be required in future versions of this PP.

For the second selection in FCS_RBG_EXT.1.1, the ST author should select the standard to which the RBG services comply (either 800-90 or 140-2 Annex C).

SP 800-90A contains four 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 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 are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CTR_DRBG are allowed. While any of the curves defined in 800-90A are allowed for Dual_EC_DRBG, the ST author not only must include the curve chosen, but also the hash algorithm used.

For the first selection in FCS_RBG_EXT.1.2, the ST author indicates whether the sources of entropy are software-based or platform-based, or both. If there are multiple sources of entropy, the ST will describe each entropy source and whether it is software- or platform-based. Platform-based noise sources are preferred.

The platform-based RBG source is the output of a validated RBG provided by the platform, which is used as an entropy source for a TSF-provided DRBG according to FCS_RBG_EXT.1.1. In this way, the developer has chained RBGs as described in NIST SP800-90C.

Note that for FIPS Pub 140-2 Annex C, currently only the method described in NIST-Recommended Random Number Generator Based on ANSI X9.31 Annex A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section 3 is valid. If the key length for the AES implementation used here is different than that used to encrypt the user data, then FCS_COP.1 may have to be adjusted or iterated to reflect the different key length. For the selection in FCS_RBG_EXT.1.2(1), the ST author selects the minimum number of bits of entropy that is used to seed the RBG.

The ST author also ensures that any underlying functions are included in the baseline requirements for the TOE.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the RBG functions claimed in that platform's ST contains the RBG functions in the Email Client's ST. The evaluator shall also examine the TSS of the Email Client's ST to verify that it describes (for each supported platform) how the RBG functionality is invoked for each operation they are used for in the Email Client (it should be noted that this may be through a mechanism that is not implemented by the Email Client; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the Email Client

Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Annex E.

If the ST author has selected a platform-based noise source, the evaluator shall verify that the platform's RBG has been validated by examining the platform's ST. The evaluator shall verify that the platform's RBG is seeded with at least the amount of entropy selected by the ST author for this profile. In this case, the ST author is not responsible for Annex E documentation of the platform's RBG.

The evaluator shall also 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 evaluator 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.

The evaluator shall perform a Variable Seed Test. The evaluator shall provide a set of 128 (Seed, DT) pairs to the TSF RBG function, each 128 bits. The evaluator 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 evaluator ensures that the values returned by the TSF match the expected values.

The evaluator 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 evaluator shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluator then invokes 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 Annex A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section 3. The evaluator ensures that the 10,000th value produced matches the expected value.

Implementations Conforming to NIST Special Publication 800-90A

The evaluator shall perform 15 trials for the RBG implementation. If the RBG 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 RBG functionality.

If the RBG 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 RBG 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 df 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 <= 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.

4.3.1.4 Secure/Multipurpose Internet Mail Extensions (S/MIME)

FCS_SMIME_EXT.1.1 The [selection: TSF, TOE platform] shall implement both a sending and receiving S/MIME v3.2 Agent as defined in RFC 5751, using CMS as defined in RFCs 5652, 5754, and 3565.

Application Note: *The RFCs allow for an agent to be either sending or receiving, or to include both capabilities. The intent of this requirement is to ensure that the TOE is capable of both sending and receiving S/MIME v3.2 messages.*

FCS_SMIME_EXT.1.2 The [selection: TSF, TOE platform] shall transmit the ContentEncryptionAlgorithmIdentifier for AES-128 CBC and [selection: AES-256 CBC, no others].

Application Note: *AES was added to CMS as defined in RFC 3565.*

FCS_SMIME_EXT.1.3 The [selection: TSF, TOE platform] shall present the digestAlgorithm field with the following Message Digest Algorithm identifiers [selection: id-sha256, id-sha384, id-sha512] and no others.

FCS_SMIME_EXT.1.4 The [selection: TSF, TOE platform] shall present the AlgorithmIdentifier field with the following sha256withRSAEncryption and [selection: sha384WithRSAEncryption, sha512WithRSAEncryption, ecdsa-with-SHA256, ecdsa-with-sha384, ecdsa-with-sha512] and no other algorithms.

Application Note: *RFC 5751 mandates that receiving and sending agents support RSA with SHA256. The algorithms to be tested in the evaluated configuration are limited by this requirement. Any other algorithms implemented that do not comply with these requirements should not be included in an evaluated TOE. The ECDSA algorithms listed above are preferred for implementation and will be mandated in future versions of this PP.*

FCS_SMIME_EXT.1.5 The [selection: TSF, TOE platform] shall use separate private keys (and associated certificates) for signature and for encryption.

FCS_SMIME_EXT.1.6 The [selection: TSF, TOE platform] shall only accept a signature from a certificate with the digitalSignature bit set.

Application Note: *It is acceptable to assume that the digitalSignature bit is set in cases where there is no keyUsage extension.*

FCS_SMIME_EXT.1.7 The [selection: TSF, TOE platform] shall implement mechanisms for retrieval of Certificate Revocation Lists (CRLs) and certificates [selection: each time needed, for incoming/outgoing messages, periodically].

Application Note: *The TOE can define how this mechanism behaves, but it is required that a mechanism exist such that revocation status is supported via CRLs and so that certificates can be retrieved for sending/receiving messages. In this requirement "periodically" can be interpreted as a one-time function with local storage, as a regularly scheduled retrieval, or as a mechanism that requires manual intervention. If the retrieval mechanism is periodic in nature,* *then the ST author will need to include an iteration of FCS for storage of CRLs; storage of certificates is covered in FCS_CKM. The import of certificates and certificate chains is not included in this requirement, but is covered in FIA_X509 and FMT_MOF.*

Assurance Activity:

The evaluator shall check the description of the implementation of S/MIME in the TSS to ensure that the proper version is specified. In addition, the evaluator shall verify that the algorithms supported are specified, and that the algorithms specified are those listed for this component. The evaluator shall also review the Operational Guidance to ensure that it contains instructions on configuring the TOE such that it complies to the description in the TSS.

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

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

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

The evaluator shall verify that the TSS describes the retrieval mechanisms for both CRLs and certificates as well as the frequency at which these mechanisms are implemented. If the TSS indicates that the mechanisms are configurable, the evaluator shall verify that the AGD guidance includes the configuration of these mechanisms.

The evaluator shall perform the following tests:

These tests can be performed in conjunction with the tests defined in FIA_X509 for certificate/certificate chain verification.

Test 1: The evaluator shall both send and receive a message with no protection (no signature or encryption) and verify that the message is transmitted properly and can be viewed at the receiving agent. This transmission can be performed as part of a number of mechanisms; it is sufficient to observe that the message arrives at the intended recipient with the same content as when sent.

Test 2: The evaluator shall both send and receive a signed message using each of the algorithms specified in the ST corresponding to the requirement. In addition, the evaluator shall use a man-in-the-middle tool to modify at least one byte of the message such that the signature is no longer valid. This can be done by modifying the content of the message over which the signature is calculated or by modifying the signature itself. The evaluator shall verify that the received message fails the signature validation check.

Test 3: The evaluator shall both send and receive an encrypted message using each of the algorithms specified in the ST. S/MIME is intended for end-to-end security. As a result, it is not possible to have a third party inspecting email and also have secure end-to-end communications. Therefore, it is not necessary to examine the encrypted traffic in an attempt to determine the ciphersuite and the proper implementation. In addition, the evaluator shall use a *man-in-the-middle tool to modify at least one byte of the message such that the encryption is no longer valid. The evaluator shall verify that the received message fails to decrypt.*

Test 4: The evaluator shall both send and receive a message that is both signed and encrypted. It is not necessary to examine the encrypted traffic to meet the intent of the requirement. In addition, the evaluator shall use a man-in-the-middle tool to modify at least one byte of the message such that the encryption and signature are no longer valid. The evaluator shall verify that the received message fails to decrypt, fails the signature validation check, and/or both.

Test 5: The evaluator shall send a signed message to the TOE using a signature algorithm not supported according to the digestAlgorithm ID (e.g., SHA1). The evaluator shall verify that the TOE does not accept the message.

Test 6: The evaluator shall send an encrypted message to the TOE using an encryption algorithm not supported according to the AlgorithmIdentifier field. The evaluator shall verify that the TOE does not accept the message.

Test 7: The evaluator shall send the TOE a message signed by a certificate without the digitalSignature bit set. The evaluator shall verify that the TOE notifies the user that the signature is invalid.

Test 8: The evaluator shall send the TOE a message signed by a certificate without the Email Protection purpose in the extendedKeyUsage. The evaluator shall verify that the TOE notifies the user that the signature is invalid.

4.3.1.5 Transport Layer Security (TLS)

FCS_TLSC_EXT.1.1 The [selection: TSF, TOE platform] shall implement TLS 1.2 (RFC 5246) supporting the following ciphersuites: [

- Mandatory Ciphersuites:
  - TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
  - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 6460
  - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 6460
- [selection: Optional Ciphersuites:
  - TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268
  - TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
  - TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268
  - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492
  - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492
  - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492
  - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492
  - TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246
  - TLS_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246
  - TLS_DHE_RSA_WITH_AES_128_CBC_ SHA256 as defined in RFC 5246
  - TLS_DHE_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246
  - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
  - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
  - no other ciphersuite]].

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

FCS_TLSC_EXT.1.2 The <u>[selection: TSF, TOE platform]</u> shall not establish a trusted channel if the distinguished name (DN) contained in a certificate does not match the expected DN for the peer.

Application Note: The DN may be in the Subject Name field or the Subject Alternative Name extension of the certificate. The expected DN may either be configured or may be compared to the Domain Name or IP address used by the peer.

Trusted communication channels include any of TLS, HTTPS, or S/MIME performed by the TSF or TOE platform. Validity checking to establish the trusted channel is performed in conjunction with FIA_X509_EXT.1.

FCS_TLSC_EXT.1.3 The <u>[selection: TSF, TOE platform]</u> shall present the signature_algorithm extension in the Client Hello with the following hash algorithms: [selection: SHA256, SHA384, SHA512] and no other hash algorithms.

Application Note: This requirement limits the hashing algorithms supported for the purpose of digital signature verification by the 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.

FCS_TLSC_EXT.1.4 The <u>[selection: TSF, TOE platform]</u> shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [selection: secp256r1, secp384r1, secp521r1] and no other curves.

Application Note: This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from **FCS_COP.1(3)** and **FCS_CKM.1(**
1) and **FCS_CKM.1(2)**. This extension is required for clients supporting Elliptic Curve ciphersuites.

Authors' Note: The iterations of the FCS requirements above vary in each PP. These numbers come from the Mobile Device PP and indicate the requirements for digital signature generation/verification, key establishment key generation, and authentication key generation.

Assurance Activity:

The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component. 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 verify that the TSS describes how the DN in the certificate is compared to the expected DN. If the DN is not compared automatically to the Domain Name or IP address, the evaluator shall ensure that the AGD guidance includes configuration of the expected DN for the connection.

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 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 establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).
- Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.
- Test 3: The evaluator shall attempt a connection with a certificate where the DN matches either the configured expected DN or the Domain Name/IP address of the peer. The evaluator shall verify that the TSF is able to successfully connect. The evaluator shall attempt a connection with a certificate where the DN does not match either the configured expected DN or the Domain Name/IP address of the peer. The evaluator shall verify that the TOE is not able to successfully connect. A user notification indicating the failure of the connection is acceptable in accordance with FIA_X509_EXT.2.3.
- Test 4: The evaluator shall configure the server to send a certificate in the TLS connection that is not supported according to the Client's signature_algorithm 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.
- Test 5: The evaluator shall configure the server to perform an ECDHE key exchange in the TLS connection using a non-supported curve (for example P-192) and shall verify

that the TOE disconnects after receiving the server's Key Exchange handshake message.

- Test 6: The evaluator shall configure the server to send a certificate in the TLS connection that the does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA certificate while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.
- Test 7: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection.
- Test 8: The evaluator shall setup a man-in-the-middle tool between the TOE and the server and shall perform the following modifications to the traffic:
  - Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.
  - Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.
  - Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.
  - Modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange.
  - 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 that the server rejects the connection after receiving the Client Finished handshake message.
  - Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.
  - Send an unencrypted packet from the Server after the client has issued the ChangeCipherSpec message and verify that the client denies the connection.

4.3.2 Class: Identification and Authentication (FIA)

4.3.2.1 S/MIME

FIA_SMIME_EXT.1.1 The [selection: TSF, TOE platform] shall only send/receive messages using the cryptographic algorithms defined in FCS_SMIME_EXT.1. The TSF shall not allow for weaker algorithms to be used for encryption.

Application Note: *While S/MIME v.3.2 requires default algorithms, these default algorithms are substantially weaker than required by this PP. This requirement indicates that the evaluated configuration of the client must disable these weaker algorithms.*

Assurance Activity:

The TSS description and operational guidance associated with this requirement are performed in conjunction with FCS_SMIME_EXT.1.

The evaluator shall perform the following test:

Test 1: The evaluator shall attempt to send the client a message signed and encrypted with disallowed algorithms and shall verify that the client rejects the message.

4.3.2.2 SASL

FIA_SASL_EXT.1.1 The [selection: TSF, TOE platform] shall implement support for Simple Authentication and Security Layer (SASL) that complies with RFC 4422.

Assurance Activity:

The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to Mail User Agent and a Mail Transfer Agent in terms of the SASL connection, along with TOE-specific options or procedures that might not be reflected in the specification.

The evaluator shall confirm that the operational guidance contains instructions for establishing the connection to the Mail User Agent and the Mail Transfer Agent. The evaluator shall also perform the following test:

Test 1: The evaluators shall ensure that the TOE is able to initiate communications with a Mail User Agent using POP, IMAP, and any assigned protocols specified in the requirement, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Test 2: The evaluators shall verify that a successful SASL handshake takes place. To perform this test, the evaluator shall use a sniffer and a packet analyzer. The packet analyzer must indicate that the protocol in use is SASL.

FIA_SASL_EXT.1.2 The [selection: TSF, TOE platform] must support the POP3 CAPA and AUTH extensions for the SASL mechanism.

FIA_SASL_EXT.1.3 The [selection: TSF, TOE platform] must support the IMAP CAPABILITY and AUTHENTICATE extensions for the SASL mechanism.

FIA_SASL_EXT.1.4 The [selection: TSF, TOE platform] must support the SMTP AUTH extension for the SASL mechanism.

Application Note: *In order for an email clients to support PKI X.509 Certificates for POP3, IMAP and SMTP as required in this document, the client must support the Simple Authentication and Security Layer (SASL) authentication method as described in RFC 4422, the AUTH and CAPA extensions for POP3, as described in RFC 5043, the AUTHENTICATION and CAPABILITY extensions for IMAP, as described in RFC 4959 and the AUTH extension for SMTP, as described in RFC 4954.*

Assurance Activity:

The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to a Mail User Agent and a Mail Transfer Agent in terms of the SASL connection, along with TOE-specific options or procedures that might not be reflected in the specification.

The evaluator shall confirm that the operational guidance contains instructions for establishing the connection to the Mail User Agent and the Mail Transfer Agent. The evaluator shall also perform the following tests:

Test 1: The evaluators shall ensure that the TOE is able to initiate communications with a Mail User Agent using POP, IMAP and SMTP and requiring SASL, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Test 2: The evaluator shall ensure, for each communication channel with an authorized IT entity in tests 1, that a valid SASL handshake is performed. To perform this test, the evaluator shall use a sniffer and a packet analyzer. The packet analyzer must indicate that the protocol in use is SASL.

4.3.2.3 X.509 Validation

FIA_X509_EXT.1 Extended: X509 Validation

FIA_X509_EXT.1.1 The [selection: *TSF, TOE platform*] shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The TSF 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 TSF shall validate the revocation status of the certificate using a Certificate Revocation List (CRL) as specified in RFC 5759 and [selection: *the Online Certificate Status Protocol (OCSP) as specified in RFC 2560, no other revocation status checking mechanism*].
- The TSF 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).
    - 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.
    - 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.

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. Certificates may optionally be used for trusted updates of the TOE (FPT_TUD_EXT.1.3) and installation of plug-ins and extensions (FPT_TUD_EXT.1.2) and, if implemented by the TOE, must be validated to contain the Code Signing purpose extendedKeyUsage. Certificates must be used to perform authentication with Web Servers using FCS_TLSC_EXT.1 and must be validated to contain the Server Authentication purpose extendedKeyUsage. Certificates must be used for S/MIME email *encryption and signature using FCS_SMIME_EXT.1 and must be validated to contain the Email Protection purpose extendedKeyUsage.*

Regardless of the selection of TSF or TOE platform, the validation is expected to end in a trusted root certificate in a root store managed by the platform.

While FIA_X509_EXT.1.1 requires that the TOE perform certain checks on the certificate presented by a TLS server, there are corresponding checks that the server will have to perform on a certificate presented by the client; namely that the extendedKeyUsage field of the client certificate includes "Client Authentication", that the digital signature bit is set, and that the key agreement bit (for the Diffie-Hellman ciphersuites) or the key encipherment bit (for RSA ciphersuites) be set. Certificates obtained for use by the TOE will have to conform to these requirements in order to be used in the enterprise.

While FIA_X509_EXT.1.1 requires that the TOE perform certain checks on the certificates presented for S/MIME, there are corresponding checks that the sender/recipient will have to perform on a certificate presented by the client; namely that the extendedKeyUsage field of the client certificate includes "Email Protection", that the digital signature bit (for S/MIME signatures) is set, and that the key agreement bit (for Diffie-Hellman encryption schemes) or the key encipherment bit (for RSA encryption schemes) be set. Certificates obtained for use by the TOE will have to conform to these requirements in order to be used in the enterprise.

FIA_X509_EXT.1.2 The [selection: *TSF, TOE platform*] shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

Application Note:

This requirement applies to certificates that are used and processed by the Email Client or platform.

Assurance Activity:

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.

The tests described must be performed in conjunction with the other Certificate Services assurance activities, including the use cases in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules.

Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function (application validation, trusted channel setup, or trusted software update) failing. 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 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 or OCSP is selected; if both are selected, and then a test is performed for each method. The evaluator has to only test one up in the trust chain (future revisions may require to ensure the validation is done up the entire chain). 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 will be revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.

Test 4: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate does not contain the basicConstraints extension. The validation of the certificate path fails.

Test 5: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension not set. The validation of the certificate path fails.

Test 6: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeds.

Test 7: The evaluator shall modify a single byte in the middle of the certificate and demonstrate that the certificate fails to validate.

4.3.2.4 X.509 Authentication and Encryption

FIA_X509_EXT.2 Extended: X509 Authentication and Encryption

FIA_X509_EXT.2.1(1) The [selection: *TSF, TOE platform*] shall use X.509v3 certificates as defined by RFC 5280 to support authentication for TLS.

FIA_X509_EXT.2.1(2) The [selection: *TSF, TOE platform*] shall use X.509v3 certificates as defined by RFC 5280 to support encryption and authentication for S/MIME.

FIA_X509_EXT.2.1(3) The [selection: *TSF, TOE platform*] shall use X.509v3 certificates as defined by RFC 5280 to support code signing for TOE updates and [selection: *code signing for extension installation, code signing for plug-in installation, no additional uses*].

Application Note:

Certificates must be used for trusted updates of TOE software and may optionally be used for installation of plug-ins and extensions.

FIA_X509_EXT.2.2 When the [selection: *TSF, TOE platform*] cannot establish a connection to determine the validity of a certificate, the [selection: *TSF, TOE platform*] shall [selection: *allow the administrator to choose whether to accept the certificate in these cases, accept the certificate, not accept the certificate*].

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 second 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. If the administrator-configured option is selected by the ST Author, the ST Author must also select function 10 in FMT_SMF.1.*

FIA_X509_EXT.2.3 The [selection: *TSF, TOE platform*] shall not establish a trusted communication channel if the peer certificate is deemed invalid.

Application Note:

Trusted communication channels include any of TLS performed by the TSF. Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280.

FIA_X509_EXT.2.4 The [selection: TSF, TOE platform] shall not install code if the code signing certificate is deemed invalid.

Application Note:

Certificates may optionally be used for trusted updates of system software (FPT_TUD_EXT.1.3).

FIA_X509_EXT.2.5 The [selection: TSF, TOE platform] shall not encrypt email if the email protection certificate is deemed invalid.

FIA_X509_EXT.2.6 The [selection: TSF, TOE platform] shall notify the user if an email is signed with an invalid certificate.

Assurance Activity:

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 and protecting email. 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.

The evaluator shall perform Test 1 for each function listed in FIA_X509_EXT.2.1 that requires the use of certificates:

Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing or in the receipt of a user notification (as required by elements 3, 4, 5, and 6). The evaluator shall then load into the platform's root store any certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds or results in no user notification.

Test 2: 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.*

4.3.3 Class: Security Management (FMT)

4.3.3.1 Management of Functions in TSF

FMT_MOF.1.1(1) The TSF and [selection: TOE platform, no other mechanism] shall restrict the ability to perform the functions:

1. Enable/disable downloading embedded objects globally and [selection: by domain, by sender, no other method]
2. Enable/disable plaintext only mode globally and [selection: by domain, by sender, no other method]
3. Enable/disable rendering and execution of attachments globally and [selection: by domain, by sender, no other method]

[selection:

4. Enable/disable email notifications,
5. Configure certificates for S/MIME use,
6. Configure certificates for TLS use,
7. Enable/disable display of email contents in email notifications,
8. Configure a certificate repository for encryption,
9. Configure whether to establish a trusted channel or disallow establishment if the TSF cannot establish a connection to determine the validity of a certificate
10. No other management functions]

to the administrator according the administrator policy.

Application Note: *The Administrator is responsible for management activities, including setting the policy that is applied by the enterprise on the email client. This administrator is likely to be acting remotely and could be the MTA administrator acting through a centralized management console or dashboard.*

The intent of this requirement is to allow the administrator to configure the TOE with a policy that may not be over-ridden by the user. If the administrator has not set a policy for a particular function, the user may still perform that function. Enforcement of the policy is done by the TOE itself, or the TOE and the TOE platform in coordination with each other.

For Function #9 per FIA_X509_EXT.2.2, the administrator has the option of accepting or rejecting all certificates that cannot be validated, accepting a given certificate that cannot be validated, or not accepting a given certificate that cannot be validated. Depending on the choice that the administrator has made in FIA_X509_EXT.2.2, the trusted connection will either be allowed for all certificates that cannot be validated, disallowed for all certificates cannot be validated, allowed for a given certificate that cannot be validated, or disallowed for a given certificate that cannot be validated.

Assurance Activity:

The evaluator shall perform the following tests:

Test 1: The evaluator shall use the test environment to deploy policies to the email client.

Test 2: The evaluator shall create policies which collectively include all management functions which are controlled by the (enterprise) administrator and cannot be overridden by the user as defined in FMT_MOF.1.1(2). The evaluator shall apply these policies to the client, attempt to override each setting as the user, and ensure that the TSF does not permit it.

4.3.3.2 Specification of Management Functions

FMT_SMF.1.1 The TSF and [selection: TOE platform, no other mechanism] shall be capable of the following management functions:

1. Enable/disable email notifications
2. Enable/disable downloading embedded objects globally and [selection: by domain, by sender, no other method]
3. Enable/disable plaintext mode globally and [selection: by domain, by sender, no other method]
4. Enable/disable rendering and execution of attachments globally and [selection: by domain, by sender, no other method]
5. Configure certificates for S/MIME use
6. Configure the option to automatically sign emails with S/MIME across the enterprise
7. Configure the option to automatically encrypt emails with S/MIME across the enterprise
8. Configure certificates for TLS use

[selection:

9. Enable/disable display of email contents in email notifications
10. Configure a certificate repository for encryption.
11. Configure whether to establish a trusted channel or disallow establishment if the TSF cannot establish a connection to determine the validity of a certificate,
12. No other management functions]

Application Note: *The management functions above specify security management functionality.*

Management functions 3, 4, and 5 must be configurable as global rule for emails processed by the client but may also be set on a per-domain or a per-sender basis.

Management of embedded objects must include all HTML and JavaScript content.

Plaintext mode disables rendering and execution of embedded objects.

Certificate repositories may be managed by the platform or by the TOE according to FDP_SMIME_EXT.1.2. If certificate repositories are managed by the TOE, function 10 must be included in the ST. Certificate repositories may be local or remote. Remote certificate repositories are a directory server, such as one running a LDAP service; thus, configuration of a remote certificate repository involves providing connection information for the server.

Function 11 must be included if the administrator-configured selection is made in FIA_X509_EXT.2.2. That is to say, per FIA_X509_EXT.2.2, the administrator has the option of *accepting or rejecting all certificates that cannot be validated, accepting a given certificate that cannot be validated, not accepting a given certificate that cannot be validated. Depending on the choice that the administrator has made in FIA_X509_EXT.2.2, the trusted connection will either be allowed for all certificates that cannot be validated, disallowed for all certificates cannot be validated, allowed for a given certificate that cannot be validated, or disallowed for a given certificate that cannot be validated.*

Assurance Activity:

The evaluator shall verify that the TSS describes each function and indicates whether the user or administrator or both may perform the management function.

The evaluator shall consult the AGD guidance to perform each of the following tests, iterating each test as necessary if both the user and administrator may perform the function. The following test numbers correspond to the function numbers:

Test 1: The evaluator shall enable email notifications and verify that the notification occurs in conjunction with the test for FDP_NOT_EXT.1. The evaluator shall then disable the email notifications, send the client an email, and verify that the notification does not occur.

Test 2: The evaluator shall send the client mail containing an embedded object that requires downloading from an external server (for example, an image) and verify that the image downloads while downloading is enabled. The evaluator shall disable downloading, send the email again, and verify that the object does not download.

Test 3: The test of this function is performed in conjunction with FDP_REN_EXT.1.1.

Test 4: The evaluator shall send the TOE an image attachment and verify that it renders. The evaluator shall then disable rendering and verify that the image does not render. The evaluator shall send the TOE an executable attachment and verify that it may be executed. The evaluator shall disable execution of attachments and then verify that the attachment cannot be executed.

Test 5: The evaluator shall configure a certificate for use with S/MIME and shall perform the tests associated with FCS_SMIME_EXT.1. The evaluator shall remove the certificate, repeat the tests to sign and decrypt a message, and verify that they fail.

Test 6: The test of this function is performed in conjunction with FCS_SMIME_EXT.1.

Test 7: The test of this function is performed in conjunction with FCS_SMIME_EXT.1.

Test 8: The evaluator shall configure the TOE and test environment server to use mutually authenticated TLS and shall configure a certificate on the TOE for this purpose. The evaluator shall cause the TOE to initiate communication via TLS and verify that the connection can be established. The evaluator shall use a packet analyzer to verify that the certificate configured is used to authenticate the connection.

Test 9: (Conditional) The evaluator shall send the TOE an email and verify that the email contents are contained in the notification. The evaluator shall disable the display of email contents in notifications, repeat the test, and verify that the email contents are not contained in the notification.

10: (Conditional) The evaluator shall configure the certificate repository according to the AGD guidance. If the TOE has a local repository, the evaluator shall load a certificate for a recipient, attempt to send an encrypted email to the recipient, and verify that it succeeds. The evaluator shall then remove the certificate, attempt to send the encrypted email, and verify that the attempt fails. If the TOE has a remote repository, the evaluator shall configure the remote repository, attempt to send an encrypted email to a user in the repository, and verify that the *attempt succeeds. The evaluator shall then remove the connection to the repository, attempt to send an encrypted email, and verify that the attempt fails.*

Test 11: (Conditional) The test of this function is performed in conjunction with FIA_X509_EXT.2.2.

4.3.3.3 FMT_SMR.1 Security Management Roles

FMT_SMR.1.1 The [selection: TOE, TOE platform] shall maintain the roles: *Administrator*.

FMT_SMR.1.2 The [selection: TOE, TOE platform] shall be able to associate users with roles.

Assurance Activity:

TSS

The evaluator shall examine the TSS and user documents to verify that they describe the administrator role and the powers granted to and limitations of the role.

Guidance

The evaluator shall examine the operational guidance to ensure that it contains instructions for administering the TOE and which interfaces are supported.

Tests

In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of administering the TOE that conforms to the requirements of this PP be tested; for instance, if the TOE can be administered through a local hardware interface or TLS/HTTPS then both methods of administration must be exercised during the evaluation team's test activities.

4.3.4 Class: Protection of the TSF (FPT)

4.3.4.1 TSF Self-Test

FPT_TST_EXT.1 Extended: TSF Self-Test

FPT_TST_EXT.1.1 The [selection: TOE, TOE platform] shall run a suite of self-tests during initial start-up (on power on) to ensure the integrity of its executable and data.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure that it specifies the self-tests that are performed at start-up; this description should include an outline of what the tests are actually doing (e.g., verifying the integrity of the TOE executable). The TSS must include any error states that the TSF or TOE platform may enter when self-tests fail, and the conditions and actions necessary to exit the error states and resume normal operation. The evaluator shall verify that the TSS indicates these self-tests are run at start-up automatically, and do not involve any inputs from or actions by the user or operator.

Guidance

N/A

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall perform the integrity check on a known good TSF executable and verify that the check is successful.
- Test 2: The evaluator shall modify the TOE executable, performs the integrity check on the modified TSF executable and verify that the check fails.
- Test 3: The evaluator shall perform the integrity check on known good TOE data and verify that the check is successful.
- Test 4: the evaluator shall modify the TOE configuration data, perform the integrity check on the modified TOE data, and verify that the check fails.

4.3.4.2 Trusted Update

FPT_TUD_EXT.1 Extended: Trusted Software Updates and Patches

FPT_TUD_EXT.1.1 The [selection: <u>TOE, TOE platform</u>] shall provide the ability to query the current version of the TOE software, [selection: <u>extension, plug-in, no other add-on</u>].

FPT_TUD_EXT.1.2 The [selection: <u>TOE, TOE platform</u>] shall provide the ability to initiate updates and patches to the TOE software, [selection: <u>extension, plug-in, no other add-on</u>].

FPT_TUD_EXT.1.3 The [selection: <u>TOE, TOE platform</u>] shall provide a means to verify software updates and patches to the TOE, [selection: <u>extension updates, plug-in updates, no other updates</u>] using a digital signature mechanism and [selection: <u>published hash, no other functions</u>] prior to installing those updates and patches.

FTP_TUD_EXT.1.4 The [selection: <u>TOE, TOE platform</u>] shall provide the ability to install TOE updates and patches, [selection: <u>extensions, plug-ins, no other add-on</u>] automatically after verification.

Application Note:

The digital signature mechanism referenced in the third element is the one specified in FCS_COP.1(3). The published hash referenced is generated by one of the functions specified in FCS_COP.1(2).

If extensions or plug-ins are selected above, the applicable selection-based requirements from Annex C must also be included in the main body of the ST.

Assurance Activity:

TSS

Updates to the TOE are signed by an authorized source and may also have a hash associated with them. The definition of an authorized source must be contained in the TSS, along with a description of how the certificates used by the update verification mechanism are contained on the system. The evaluator shall ensure this information is contained in the TSS.

The evaluator shall also ensure that the TSS (or the operational guidance) describes how the candidate updates are obtained; the processing associated with verifying the digital signature or calculating the hash of the updates; and the actions that take place for successful (hash or signature was verified) and unsuccessful (hash or signature could not be verified) cases. If these activities are performed entirely by the underlying platform, a reference to the ST of each platform indicating that the required functionality is included for each platform shall be verified by the evaluator.

Guidance

The evaluator shall examine the operational guidance to verify it documents the steps for verifying the current version of the TOE software, initiating updates and patches to the TOE software, and configuring verification of software updates and patches.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall perform the version verification activity to determine the current version of the product. The evaluator shall obtain a legitimate update using procedures described in the operational guidance and verify that it is successfully installed on the TOE. Then, the evaluator shall perform a subset of other assurance activity tests to demonstrate that the update functions as expected. After the update, the evaluator shall perform the version verification activity again to verify the version correctly corresponds to that of the update.
- Test 2: The evaluator shall perform the version verification activity to determine the current version of the product. The evaluator shall obtain or produce an illegitimate update, and attempt to install it on the TOE or TOE platform. The evaluator shall verify that the TOE or TOE platform rejects the update.
- Test 3: The evaluator shall attempt to install an unsigned patch or update and shall verify that installation fails.
- Test 4: the evaluator shall sign a patch or update with an invalid certificate. The evaluator shall attempt to install the patch or update and shall verify that installation fails.
- Test 5: The evaluator shall sign a patch or update with a certificate containing a missing or invalid code signing entendedKeyUsage extension. The evaluator shall attempt to install the patch or update and shall verify that installation fails.
- Test 6: The evaluator shall attempt to install a signed patch or update and shall verify that installation is successful and happens automatically after the signature is verified.

4.3.5 Class: Trusted Path/Channels (FTP)

4.3.5.1 Trusted Channel Communication

FTP_ITC.1 Inter-TSF Trusted Channel

FTP_ITC.1.1 The [selection: TSF, TOE platform] shall use TLS to provide a trusted communication channel between itself and authorized IT entities supporting the following capabilities [selection: directory server, administrative configuration server, no other entities], that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data.

FTP_ITC.1.2 The [selection: TSF, TOE platform] shall permit the TSF to initiate or receive communication via the trusted channel.

FTP_ ITC.1.3 The [selection: TSF, TOE platform] shall initiate communication via the trusted channel for [selection: *IMAP, SMTP, POP,MAPI Extensions for HTTP, MAPI/RPC, ActiveSync*].

Assurance Activity:

The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to Mail User Agent and a Mail Transfer Agent in terms of the TLS connection, along with TOE-specific options or procedures that might not be reflected in the specification.

The evaluator shall confirm that the operational guidance contains instructions for establishing the connection to the Mail User Agent and the Mail Transfer Agent. The evaluator shall also perform the following tests:

Test 1: The evaluators shall ensure that the TOE is able to initiate communications with a Mail User Agent using POP, IMAPand any assigned protocols specified in the requirement over TLS, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Test 2: The evaluators shall ensure that the TOE is able to initiate communications with a Mail Transfer Agent using SMTP and any assigned protocols specified in the requirement over TLS, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity in tests 1 and 2, the channel data is not sent in plaintext. To perform this test, the evaluator shall use a sniffer and a packet analyzer. The packet analyzer must indicate that the protocol in use is TLS.

5 Security Assurance Requirements

The Security Objectives for the TOE in Section 5.4 were constructed to address threats identified in Section 5.2. The Security Functional Requirements (SFRs) in Section 4.2 are a formal instantiation of the Security Objectives. The PP draws from the CC Security Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

While this section contains the complete set of SARs from the CC, the Assurance Activities to be performed by an evaluator are detailed both in Section 5.2 as well as in this section.

The general model for evaluation of TOEs against STs written to conform to this PP is as follows:
After the ST has been approved for evaluation, the Common Criteria Testing Laboratory (CCTL) will obtain access to a TOE and its supporting environmental IT, as well as its administrative guidance. The Assurance Activities listed in the ST (which will be refined by the CCTL to be TOE-specific, either within the ST or in a separate document) will then be performed by the CCTL. The results of these activities will be documented and presented (along with the administrative guidance used) for validation.

For each family, "Developer Notes" are provided on the developer action elements to clarify what, if any, additional documentation/activity needs to be provided by the developer. For the content/presentation and evaluator activity elements, additional assurance activities are described as a whole for the family, rather than for each element. Additionally, the assurance activities described in this section are complementary to those specified in Section 5.2.

The TOE security assurance requirements, summarized in Table 2, identify the management and evaluative activities required to address the threats identified in Section 5.2 of this PP.

5.1 Class ADV: Development

The information about the TOE is contained in the guidance documentation available to the end user as well as the TOE Summary Specification (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 contained in Section 4.2 should provide the ST authors with sufficient information to determine the appropriate content for the TSS section.

ADV_FSP.1 Basic functional specification

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.

Developer Note: As indicated in the introduction to this section, the functional specification is comprised of the information contained in the AGD_OPR and AGD_PRE documentation, coupled with the information provided in the TSS of the ST. The assurance 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	*The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.*
ADV_FSP.1.2C	*The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.*
ADV_FSP.1.3C	*The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.*
ADV_FSP.1.4C	*The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.*

Evaluator action elements:

ADV_ FSP.1.1E	*The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.*
ADV_ FSP.1.2E	*The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.*

Assurance Activity:

There are no specific assurance activities associated with these SARs. The functional specification documentation is provided to support the evaluation activities described in Section 4.2, 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 activities being performed; if the evaluator is unable to perform an activity because the there is insufficient interface information, then an adequate functional specification has not been provided.

5.2 Class AGD: Guidance Documents

The guidance documents will be provided with the developer's security target. Guidance must include a description of how the authorized user verifies that the Operational Environment can fulfill its role for the security functionality. The documentation should be in an informal style and readable by an authorized user.

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 TOE in that environment; and
- instructions to manage the security of the TOE as a product and as a component of the larger operational environment.

Guidance pertaining to particular security functionality is also provided; specific requirements on such guidance are contained in the assurance activities specified in Section 4.2.

AGD_OPE.1 Operational user guidance

Developer action elements:

AGD_OPE.1.1D The developer shall provide operational user guidance.

Developer Note: Rather than repeat information here, the developer should review the assurance activities for this component to ascertain the specifics of the guidance that the evaluators will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

Content and presentation elements:

AGD_OPE.1.1C The operational user guidance shall describe what the authorized user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

AGD_OPE.1.2C The operational user guidance shall describe, for the authorized user, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.3C The operational user guidance shall describe, for the authorized user, 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 The operational user guidance shall, for the authorized user, 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 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 The operational user guidance shall, for the authorized user, 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 The operational user guidance shall be clear and reasonable.

AGD_OPE.1.8C The operational user guidance shall be expressed in the eXtensible Configuration Checklist Description Format (XCCDF) to support *security automation. [Annex for US only] The operational user guidance shall express each configuration guidance item that could be used in a compliance checking regime as an XCCDF Rule element, and provide references to the NIST 800-53 controls which the item satisfies.*

Evaluator action elements:

AGD_OPE.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Assurance Activity:

Some of the contents of the operational guidance will be verified by the assurance activities in Section 4.2 and evaluation of the TOE according to the CEM.

The documentation must describe the process for verifying updates to the TOE, either by checking the hash or by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

1. *Instructions for querying the current version of the TOE software.*
2. *For hashes, a description of where the hash for a given update can be obtained. For digital signatures, instructions for obtaining the certificate that will be used by the FCS_COP.1(2) mechanism to ensure that a signed update has been received from the certificate owner. This may be supplied with the product initially, or may be obtained by some other means.*
3. *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).*
4. *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.*

AGD_PRE.1 Preparative procedures

Developer action elements:

AGD_PRE.1.1D The developer shall provide the TOE including its preparative procedures.

Developer Note: As with the operational guidance, the developer should look to the assurance activities to determine the required content with respect to preparative procedures.

Content and presentation elements:

AGD_PRE.1.1C The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer’s delivery procedures.

AGD_PRE.1.2C The preparative procedures shall describe all the steps necessary for secure installation of the TOE 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 The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_PRE.1.2E The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared securely for operation.

Assurance Activity:

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 (that is, combination of hardware and operating system) claimed for the TOE in the ST.

5.3 Class ALC: Life-cycle Support

At the assurance level provided for TOEs conformant to this PP, life-cycle support is limited to an examination of the TOE vendor's development and configuration management process. This is a result of the critical role that a developer's practices play in contributing to the overall trustworthiness of a product.

ALC_CMC.1 Labeling of the TOE

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 The TOE shall be labeled with its unique reference.

Evaluator action elements:

ALC_CMC.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Assurance Activity:

The evaluator shall verify that the TOE has been provided with its unique reference labeled. The evaluator shall verify that the CM documentation has been provided and that it describes the method used to uniquely identify each configuration item. The evaluator shall verify that the developer has used a CM system and that this system uniquely identifies each configuration item.

ALC_CMS.1 TOE CM coverage

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	*The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.*
ALC_CMS.1.2C	*The configuration list shall uniquely identify the configuration items.*

Evaluator action elements:

ALC_CMS.1.1E	*The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.*

Assurance Activity:

The evaluator shall verify that the developer has provided a configuration list for the TOE that contains each item highlighted above. The evaluator shall verify that each item in the configuration list is uniquely identified and its developer is indicated.

5.4 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 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

Testing is performed to confirm the functionality described in the TSS as well as the administrative (including configuration and operation) documentation provided. The focus of the testing is to confirm that the requirements specified in Section 4.2 are being met, although some additional testing is specified for SARs in Section 4.3. The Assurance Activities identify the additional testing activities associated with these components. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this PP.

Developer action elements:

ATE_IND.1.1D	*The developer shall provide the TOE for testing.*

Content and presentation elements:

ATE_IND.1.1C	*The TOE shall be suitable for testing.*

Evaluator action elements:

ATE_IND.1.1E	*The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.*

ATE_IND.1.2E	*The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.*

Assurance Activity:

The evaluator shall prepare a test plan and report documenting the testing aspects of the system. The test plan covers all of the testing actions contained in the CEM and the body of this PP's Assurance Activities. While it is not necessary to have one test case per test listed in an Assurance Activity, the evaluators must document in the test plan that each applicable testing requirement in the ST 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; rationale must be provided. If all platforms claimed in the ST 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 evaluators are 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) is 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/HTTPS).

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.

5.5 Class AVA: Vulnerability Assessment

168. For the first generation of this protection profile, the evaluation lab is expected to survey open sources to learn 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, evaluators 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

Developer action elements:

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

Content and presentation elements:

AVA_VAN.1.1C	*The TOE shall be suitable for testing.*

Evaluator action elements:

AVA_VAN.1.1E	*The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.*
AVA_VAN.1.2E	*The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.*
AVA_VAN.1.3E	*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 Activity:

As with ATE_IND 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 determine the vulnerabilities that have been found in MDMs in general, as well as those that pertain to the particular TOE. 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. For example, if the vulnerability can be detected by pressing a key combination on boot-up, a test would be suitable at the assurance level of this PP. 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.

6 RATIONALE

The rationale tracing the threats to the objectives and the objectives to the requirements is contained in the prose in Sections 2.0 and 3.0. The only outstanding mappings are those for the Assumptions and Organizational Security Policies; those are contained in Annex A below.

Annex A: Supporting Tables

In this Protection Profile, the focus in the initial sections of the document is to use a narrative presentation in an attempt to increase the overall understandability of the threats to network devices; the methods used to mitigate those threats; and the extent of the mitigation achieved by compliant TOEs. This presentation style does not readily lend itself to a formalized evaluation activity, so this Annex contains the tabular artifacts that can be used for the evaluation activities associated with this document.

A.1 Assumptions

The specific conditions listed in the following subsections are assumed to exist in the TOE's Operational Environment. These assumptions include both practical realities in the development of the TOE security requirements and the essential environmental conditions on the use of the TOE.

ST authors should ensure that the assumptions still hold for their particular technology; the table should be modified as appropriate.

Table 1: TOE Assumptions

Assumption Name	Assumption Definition
A.PLATFORMS	The email clients described in this document will run on any operating system, regardless of the underlying platform.
A.PHYSICAL	Physical security, commensurate with the value of the TOE and the data it contains, is assumed to be provided by the environment.
A.TRUSTED_ADMIN	TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.
A.TRUSTED_USER	The email client user is not malicious, and exercises appropriate precautions.
A.PLATFORM_FUNCTIONS	The platform supporting the email client shall offer file system and other operating system capabilities.

A.2 Threats

The following threats should be integrated into the threats that are specific to the technology by the PP authors when including the requirements described in this document. Modifications, omissions, and additions to the requirements may impact this list, so the PP author should modify or delete these threats as appropriate.

Table 2: Threats

Threat Name	Threat Definition

T.UNAUTHORIZED_ADDON	Malicious or exploitable extensions or plug-ins could be used knowingly or unknowingly by a developer, possibly resulting in the capability of attacks against the platform‘s system software.
T.UNAUTHORIZED_UPDATE	Malicious or exploitable software could be used knowingly or unknowingly by a developer, possibly resulting in the capability of attacks against the platform‘s system software.
T.NETWORK_EAVESDROP	If positioned on a wireless communications channel or elsewhere on the network, attackers may monitor and gain access to data exchanged between the email client and other endpoints
T.NETWORK_ATTACK	An attacker may initiate communications with the email client or alter communications between the email client and other endpoints.

A.3 Security Objectives for the TOE

Table 3: Security Objectives for the TOE

TOE Security Objective	TOE Objective Definition
O.COMMS	The TOE will provide the capability to communicate using one (or more) standard protocols as a means to maintain the confidentiality of data that are transmitted outside of the TOE.
O.CONFIG	The TOE will provide the capability to configure and apply security policies. This ensures the email client can protect user and enterprise data that it may store or process.
O.INTEGRITY	The TOE will provide the capability to perform self-tests to ensure the integrity of critical functionality, software/firmware and data has been maintained. The TOE will also provide a means to verify the integrity of downloaded updates, and to verify the integrity of extensions and plug-ins and that they are from a trusted source.
O.STORAGE	The TOE will provide the capability to encrypt all user and enterprise data and authentication keys to ensure the confidentiality of data that it stores.

A.4 Security Threats to Security Objectives

The following table contains a mapping of Security Threats to Objectives for the TOE.

Table 3: Security Threats to Objectives Mapping

Threat	Objective
T.UNAUTHORIZED_ADD-ON	O.INTEGRITY
T.UNAUTHORIZED_UPDATE	O.INTEGRITY
T.NETWORK_EAVESDROP	O.COMMS
T.NETWORK_ATTACK	O.COMMS; O.ISOLATION
T.DATA_ACCESS	O.STORAGE; O.CONFIG

Annex B: Optional 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. Additionally, there are three other types of requirements specified in Appendices B, C, and D.

The first type (in this Annex) are requirements that can be included in the ST, but do not have to be in order for a TOE to claim conformance to this PP. The second type (in Annex C) are requirements based on selections in the body of the PP: if certain selections are made, then additional requirements in that appendix will need to be included. The third type (in Annex D) are components that are not required in order to conform to this PP, but will be included in the baseline requirements in future versions of this PP. Note that the ST author is responsible for ensuring that requirements that may be associated with those in Annex B, Annex C, and/or Annex D but are not listed (e.g., FMT-type requirements) are also included in the ST.

At any time these may be included in the ST such that the TOE is still conformant to this PP.

B.1 Class: User Data Protection (FDP)

B.1.1 Data-at-Rest (DAR)

FDP_DAR_EXT.1 The TSF shall mark all email, persistent secrets, private keys, and [selection: calendar appointments, contacts, [assignment: other user data], no other user data] as sensitive.

Application Note: *The email client is not required to provide its own data-at-rest encryption but instead uses the Mobile Device's data-at-rest protection mechanisms to ensure that email is encrypted while the device is in the locked state. The Protection Profile for Mobile Device Fundamentals (MDFPP) defines three levels of data protection: TSF Data, Protected Data, and Sensitive Data. The intent of this requirement is that email client use the Sensitive Data level of data-at-rest protection. The MDFPP provides a requirement for the mechanism to be used by the TSF to mark data as sensitive.*

Using this mechanism, the email client must ensure that email data, persistent secrets (passwords, other credentials, secret keys) and private keys are encrypted in the locked state. Optionally, the email client may protect other user data retained by the client, including calendar appointments, contacts, and other user data (e.g. task lists, chat messages) in the same way or may use the Protected Level.

Assurance Activity:

The evaluator shall examine the TSS to ensure that it describes how the client marks the required user data as sensitive.

The developer shall provide enough source code (only a few lines are necessary) or compilation information to the evaluator to justify that user data is correctly protected by the platform. The evaluator shall examine the API documentation for the platform(s) indicated as supported by the TOE to verify that the provided source code or compilation information marking user data as sensitive matches the API guidance for the platform.

Annex C: 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 be included.

C.1 Class: User Data Protection (FDP)

C.1.1 Information Deletion

C.1.1.1 FDP_DEL_EXT.1 Extended: Deletion of Extension Information

FDP_DEL_EXT.2.1 The TOE shall provide the capability to delete extensions so that all information is removed, including extensions, configuration elements, and stored information.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure that it documents where extensions are stored, where extensions are allowed to store information.

Guidance

The evaluator shall examine the operational guidance to verify that it includes instructions for how the user can delete extensions.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall install a TOE extension, then examine the TOE’s file system to verify that the extension and extension data are stored as documented. The evaluator shall then uninstall the TOE extension and examine the TOE’s file system to verify that the extension and extension data are removed from the documented locations.

C.1.1.2 FDP_DEL_EXT.2 Extended: Deletion of Plug-in Information

FDP_DEL_EXT.3.1 The TOE shall provide the capability to delete plug-ins so that all information is removed, including plug-ins, configuration elements, and stored information.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure that it documents where plug-ins are stored, where plug-ins are allowed to store information, and whether the option exists to delete all plug-in information.

Guidance

The evaluator shall examine the operational guidance to verify that it includes instructions for how the user can delete plug-ins and associated content.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall install a TOE plug-in, then examine the TOE's file system to verify that the plug-in and plug-in data are stored as documented. The evaluator shall then uninstall the TOE plug-in and examine the TOE's file system to verify that the plug-in and plug-in data are removed from the documented locations.

C.2 Class: Protection of the TSF (FPT)

C.2.1 Trusted Update

C.2.1.1 FPT_TUD_EXT.1 Extended: Trusted Extension Update

FPT_TUD_EXT.1.1 The TOE shall provide the capability to query the current version of the extension.

FPT_TUD_EXT.1.2 The TOE shall provide the capability to initiate extension updates.

FTP_TUD_EXT.1.3 The TOE shall provide a means to verify extensions and extension updates using a digital signature mechanism prior to installing the extension or extension updates.

FTP_TUD_EXT.1.4 The TOE shall prevent the automatic installation of extensions.

Application Note:

Extensions are bundles of code that are added to the email client to add specific functionality that the client does not provide by default. Extensions could be developed by the email client vendor or by third parties and are allowed full access to view and interact with the email content that the email client sees.

Assurance Activity:

TSS

The evaluator shall examine the TSS to verify that it describes the ability of the TSF to verify that extensions and extension updates come from a trusted source. The evaluator shall examine the TSS to verify that it states that the TSF will reject extensions from unapproved sources.

Guidance

The evaluator shall examine the operational guidance to verify that it includes instructions on how to configure the TOE with trusted extension sources.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall configure the TOE with trusted sources. The evaluator shall create or obtain an extension signed by a trusted source and attempt to install it. The evaluator shall verify that the signature on the extension is valid.

- Test 2: The evaluator shall create or obtain an extension signed by an untrusted source and attempt to install it. The evaluator shall verify that the signed extension is rejected.
- Test 3: The evaluator shall create or obtain an extension signed with an invalid certificate and attempt to install it. The evaluator shall verify that the signed extension is rejected.
- Test 4: The evaluator shall create or obtain an extension signed by a trusted source, modify the extension without re-signing it, and attempt to install it. The evaluator shall verify that the signed extension is rejected.

C.2.1.2 FPT_TUD_EXT.2 Extended: Trusted Plug-in Update

FPT_TUD_EXT.2.1 The TOE shall provide the ability to query the current version of the plug-in.

FPT_TUD_EXT.2.2 The TOE shall provide the ability to initiate the downloading of plug-ins and plug-in updates.

FTP_TUD_EXT.2.3 The TOE shall provide a means to verify plug-ins using a digital signature mechanism and [selection: <u>published hash, no other functions</u>] prior to installing the plug-in.

FTP_TUD_EXT.2.4 The TOE shall prevent the automatic installation of plug-ins.

Assurance Activity:

TSS

The evaluator shall examine the TSS to verify that it states that the TSF will reject plug-ins from unapproved sources.

Guidance

The evaluator shall examine the operational guidance to verify that it includes instructions on how to configure the TOE with trusted plug-in sources.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall configure the TOE with trusted plug-in sources. The evaluator shall create or obtain a plug-in signed by a trusted source and attempt to install it. The evaluator shall verify that the signature on the plug-in is valid.
- Test 2: The evaluator shall create or obtain a plug-in signed by an untrusted source and attempt to install it. The evaluator shall verify that the signed plug-in is rejected.
- Test 3: The evaluator shall create or obtain a plug-in signed with an invalid certificate and attempt to install it. The evaluator shall verify that the signed plug-in is valid.
- Test 4: The evaluator shall create or obtain a plug-in signed by a trusted source, modify the plug-in without re-signing it, and attempt to install it. The evaluator shall verify that the signed plug-in is rejected.

Annex D: Objective 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 that specify security functionality that is desirable and these requirements are contained in this Annex. It is expected that these requirements will transition from objective requirements to baseline requirements in future versions of this PP.

D.1 Class: User Data Protection (FDP)

D.1.1 Storage of Persistent Information (FDP_PST)

FDP_PST_EXT.1.1 The TOE shall be capable of operating without storing persistent information to the file system.

Application Note: *Exceptions from this requirement are credentials and configuration information.*

Assurance Activity:

The evaluator shall operate the TOE for a period of time, ensuring that a wide variety of TOE functionality has been exercised. The evaluator shall then examine the TOE to ensure that no files have been written to the file system other than credentials or configuration information.

Annex E: Entropy Documentation and Assessment

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 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.

E.1 Design Description

Documentation shall include the design of the entropy source as a whole, including the interaction of all entropy source components. It will describe the operation of the entropy source to include how it works, 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 random comes from, where it 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.

E.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 exhibiting probabilistic behavior (an explanation of the probability distribution and justification for that distribution given the particular source is one way to describe this). This argument will include a description of the expected entropy rate and explain how you ensure 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 entropy justification shall not include any data added from any third-party application or from any state saving between restarts.

E.3 Operating Conditions

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.

E.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.

Annex F: Glossary and Acronyms

F.1 Technical Definitions

ActiveSync	Microsoft protocol for synchronizing data between mobile and desktop environments
IMAP	Internet Message Access Protocol - protocol for an email client to retrieve email from an email server over TCP/IP; IMAP4 defined in RFC 3501
MAPI	Messaging Application Programming Interface for HTTP - protocol used by Microsoft Exchange for sending/receiving email; defined in MS-OXCMAPIHTTP
MUA	Mail User Agent
MTA	Mail Transfer Agent
POP	Post Office Protocol - protocol for an email client to retrieve email from an email server over TCP/IP; POP3 defined in RFC 1939
RPC	Remote Procedure Call - Protocol used by Microsoft Exchange to send/receive MAPI commands; defined in MS-OXCRPC
SMTP	Simple Mail Transfer Protocol - protocol for an email client to send email to an email server over TCP/IP; SMTP defined in RFC 5321

F.2 Common Criteria Definitions

Assurance	Grounds for confidence that a TOE meets the SFRs [CC1].
CC	Common Criteria
PP	Protection Profile
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
Security Target (ST)	Implementation dependent statement of security needs for a specific identified TOE.

Target of Evaluation (TOE)	Set of software, firmware and hardware under evaluation, possibly accompanied by guidance.
TOE Security Functionality (TSF)	Combined functionality of all hardware, software, and firmware of a TOE that must be relied upon for the correct enforcement of the SFR
TOE Summary Specification (TSS)	Documentation which provides evaluators with a description of the implementation of SFRs in the TOE.

F.3 Acronyms

AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
CRL	Certificate Revocation List
CSP	Cryptographic Service Provider
DHE	Diffie-Hellman Key Exchange
DN	Distinguished Name
DSA	Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
FFC	Finite-Field Cryptography
FIPS	Federal Information Processing Standards
GCM	Galois/Counter Mode
HMAC	Keyed Hash Message Authentication Code
HTML	HyperText Markup Language
HTML5	HyperText Markup Language version 5
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IETF	Internet Engineering Task Force
IV	Initialization Vector

KAT	Known Answer Test
KDF	Key Derivation Function
NIST	National Institute of Standards and Technology
OCSP	Online Certificate Status Protocol
OID	Object Identifier
PDF	Portable Document Format
RFC	Request for Comment (IETF)
RSA	Rivest Shamir Adelman
SHA	Secure Hash Algorithm
TLS	Transport Layer Security