US Government Protection Profile for Single-Level Operating Systems in Medium Robustness Environments

Version 1.91

Information Assurance Directorate

16 March 2007

Foreword

1 This publication, "*U.S. Government Protection Profile for Single-level Operating Systems in Medium Robustness Environments*", is issued by the Information Assurance Directorate as part of its program to promulgate security standards for information systems. This protection profile is based on the "Common Criteria for Information Technology Security Evaluations, Version 2.3". The author elected not to include NIAP interps but included refinements as appropriate to ensure the intent of the interp was included. International interps were included in the CC version 2.3 update.

2 Further information, including the status and updates, of this protection profile can be found on the internet at: http://www.niap-ccevs.org/cc-scheme/pp/.

3 Comments on this document should be directed to: ppcomments@missi.ncsc.mil. The comments should include the title of the document, the page, the section number, and paragraph number, detailed comment and recommendations.

Table of Contents

List of Figures

List of Tables

1. Introduction

4 This section contains overview information necessary to allow a Protection Profile (PP) to be registered through a Protection Profile Registry. The PP identification provides the labeling and descriptive information necessary to identify, catalogue, register, and cross-reference a PP. The PP overview summarizes the profile in narrative form and provides sufficient information for a potential user to determine whether the PP is of interest. The overview can also be used as a stand-alone abstract for PP catalogues and registers. The "Conventions" section provides the notation, formatting, and conventions used in this protection profile. The "Glossary of Terms" section gives a basic definition of terms, which are specific to this PP. The "Document Organization" section briefly explains how this document is organized.

1.1 Identification

5 Title: U.S. Government Protection Profile for Single-level Operating Systems in Medium RobustnessVersion 1.91 Environments Version 1.91, dated 16 March 2007

6 Registration: <to be provided upon registration>

7 Keywords: operating system, COTS, medium robustness, single-level, access control, discretionary access control, DAC, cryptography

1.2 Overview

8 National Security Directive 42 delegates to NSA the authority to approve information technology products and cryptographic implementations for use in protecting national security information. This *"U.S. Government Protection Profile for Single-level Operating Systems in Medium Robustness Environments"* specifies security requirements for commercial-off-the-shelf (COTS) general-purpose operating systems in networked environments and uses Department of Defense (DoD) and National Information Assurance (IA) guidance and policies as a basis to establish the requirements for National Security Systems[1]. Products meeting this protection profile become candidates for use in National Security Systems. However, compliance to this protection profile is not, by itself, sufficient.

9 System certification and accreditation processes should consider product evaluation results when determining if acceptable levels of protection are provided. Compliance with this PP alone does not offer sufficient confidence that national security information is appropriately protected in the context of a larger system in which the TOE is integrated. Designers of such large systems must apply appropriate systems security engineering principles and defense-in-depth techniques to afford acceptable protection for national security information.

[1] National Security Systems are systems that contain classified information or involves intelligence activities, involves cryptologic activities related to national security, involves command and control of military forces, involves equipment that is an integral part of a weapon or weapon system, or involves equipment that is critical to the direct fulfillment of military or intelligence missions.

10 Conformant products support Identification and Authentication (I&A), Discretionary Access Control (DAC), an Audit Capability, and Cryptographic Services. These products provide adequate security services, mechanisms, and assurances to process unclassified information and are also candidates for processing national security information.

11 PP conformant systems are suitable for use in unclassified environments which process administrative, private, and sensitive/proprietary information and are candidates for classified environments that utilize appropriate systems engineering and defense-in-depth strategies. However, when an organization's most sensitive/proprietary information is to be sent from the TOE to another system across a publicly accessible network, the organization should also apply additional protection at the network boundaries.

1.2.1 TOE Environment Defining Factors

12 The environment for a TOE can be characterized by the authorization (or lack of authorization) of the least trustworthy entity compared to the highest value of TOE resources (i.e. the TOE itself and all of the data processed by the TOE).

13 In trying to specify the environments in which compliant TOEs are appropriate, it is useful to first discuss the two defining factors that characterize the environment: value of the resources and authorization of the entities to those resources.

14 Note that there are an infinite number of combinations of entity authorization and value of resources. In the next section 1.2.2, these two environmental factors will be related to the robustness required for selection of an appropriate TOE.

1.2.1.1 Value of Resources

15 Value of the resources associated with the TOE includes the data being processed or used by the TOE, as well as the TOE itself (for example, a real-time control processor). "Value" is assigned by the organization that owns the resources. For example, in the DoD low-value data might be equivalent to data marked "FOUO", while high-value data may be those classified Top Secret. In a commercial enterprise, low-value data might be the internal organizational structure as captured in the corporate on-line phone book, while high-value data might be corporate research results for the next generation product.

16 Note that when considering the value of the data one must also consider the value of data or resources that are accessible through exploitation of the TOE. For example, a firewall may have "low value" data itself, but it might protect an enclave with high value data. If the firewall is being depended upon to protect the high value data, then it must be treated as a high-value-data TOE.

1.2.1.2 Authorization of Entities

17 Authorization that entities (users, administrators, other IT systems) have with respect to the TOE (and thus the resources of that TOE, including the TOE itself) is an abstract concept reflecting a combination of the trustworthiness of an entity and the access and privileges granted to that entity with respect to the resources of the TOE. For instance, entities that have total authorization to all data on the TOE are at one end of this spectrum; these entities may have privileges that allow them to read, write, and modify anything on the TOE, including all TSF

data. Entities at the other end of the spectrum are those that are authorized to few or no TOE resources. For example, in the case of an operating system, an entity may not be allowed to log on to the TOE at all (that is, they are not valid users listed in the operating system's user database).

18 It is important to note that authorization does not refer to the access that the entities actually have to the TOE or its data. For example, suppose the owner of the system determines that no one other than employees is authorized to certain data on a TOE, yet they connect the TOE to the Internet. There are millions of entities that are not authorized to the data (because they are not employees), but they actually have connectivity to the TOE through the Internet and thus can attempt to access the TOE and its associated resources.

19 Entities are characterized according to the value of resources to which they are authorized; the extent of their authorization is implicitly a measure of how trustworthy the entity is with respect to any of the applicable security policies.

1.2.2 Selection of Appropriate Robustness Levels

20 Robustness is a characteristic of a TOE defining how well it can protect itself and its resources; a more robust TOE is better able to protect itself. This section relates the defining factors of IT environments, authorization, and value of resources to the selection of appropriate robustness levels.

21 When assessing any environment with respect to Information Assurance, the critical point to consider is the likelihood of an attempted security policy compromise, which was characterized in the previous section in terms of entity authorization and resource value. As previously mentioned, robustness is a characteristic of a TOE that reflects the extent to which a TOE can protect itself and its resources. It follows that as the likelihood of an attempted resource compromise increases, the robustness of an appropriate TOE should also increase.

22 It is critical to note that several combinations of the environmental factors will result in environments in which the likelihood of an attempted security policy compromise is similar. Consider the following two cases:

23 The first case is a TOE that processes only low-value data. Although the organization has stated that only its employees are authorized to log on to the system and access the data, the system is connected to the Internet to allow authorized employees to access the system from home. In this case, the least trusted entities would be unauthorized entities (e.g. non-employees) exposed to the TOE because of the Internet connectivity. However, since only low-value data are being processed, the likelihood that unauthorized entities would find it worth their while to attempt to compromise the data on the system is low and selection of a basic robustness TOE would be appropriate.

24 The second case is a TOE that processes high-value (e.g., classified) information. The organization requires that the TOE be in a closed environment, and that every user with physical and logical access to the TOE undergo an investigation so that they are authorized to the highest value data on the TOE. Because of the extensive checks done during this investigation, the organization is assured that only highly trusted users are authorized to use the TOE. In this case, even though high value information is being processed, it is unlikely that a compromise of that

data will be attempted because of the authorization and trustworthiness of the user and the closed nature of the environment; therefore, selection of a basic robustness TOE would be appropriate.

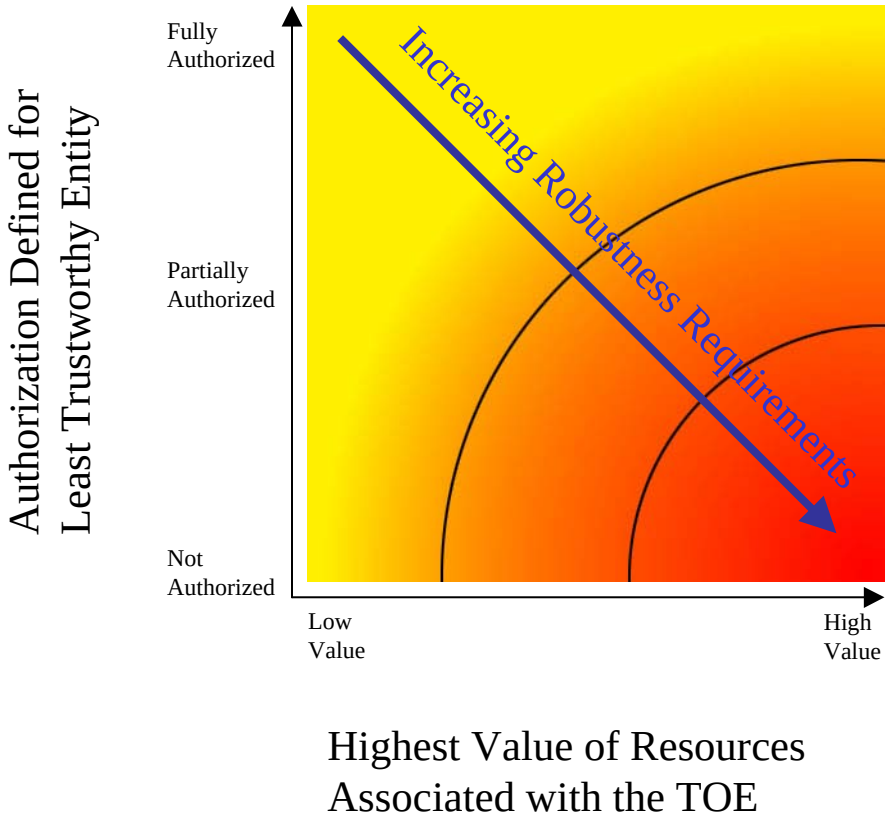

Figure 1-1 Universe of Environments

25 The preceding examples demonstrated that it is possible for radically different combinations of entity authorization and resource values to result in a similar likelihood of an attempted compromise. Figure 1-1 depicts the "universe" of environments characterized by the two factors discussed in the previous section: on one axis is the authorization defined for the least trustworthy entity, and on the other axis is the highest value of resources associated with the TOE.

26 As depicted in Figure 1-1, the robustness of the TOEs required in each environment steadily increases as one goes from the upper left of the chart to the lower right; this corresponds to the need to counter increasingly likely attack attempts by the least trustworthy entities in the environment. Note that the shading of the chart is intended to reflects the notion that different environments engender similar levels of "likelihood of attempted compromise", signified by a similar color. Further, the delineations between such environments are not stark, but rather are finely grained and gradual.

27 While it would be possible to create many different "levels of robustness" at small intervals along the "Increasing Robustness Requirements" line to counter the increasing likelihood of attempted compromise due to those attacks, it would not be practical nor particularly useful.

Instead, in order to implement the robustness strategy where there are only three robustness levels: Basic, Medium, and High, the graph is divided into three sections, with each section corresponding to a set of environments where the likelihood of attempted compromise is roughly similar. This is graphically depicted in the Figure 1-1.

28 A second representation of environments is shown in Figure 1-2, the "dots" represent given instantiations of environments; like-colored dots define environments with a similar likelihood of attempted compromise. Correspondingly, a TOE with a given robustness should provide sufficient protection for environments characterized by like-colored dots. In choosing the appropriateness of a given robustness level TOE PP for an environment, then, the user must first consider the lowest authorization for an entity as well as the highest value of the resources in that environment. This should result in a "point" in the chart above, corresponding to the likelihood that that entity will attempt to compromise the most valuable resource in the environment. The appropriate robustness level for the specified TOE to counter this likelihood can then be chosen.

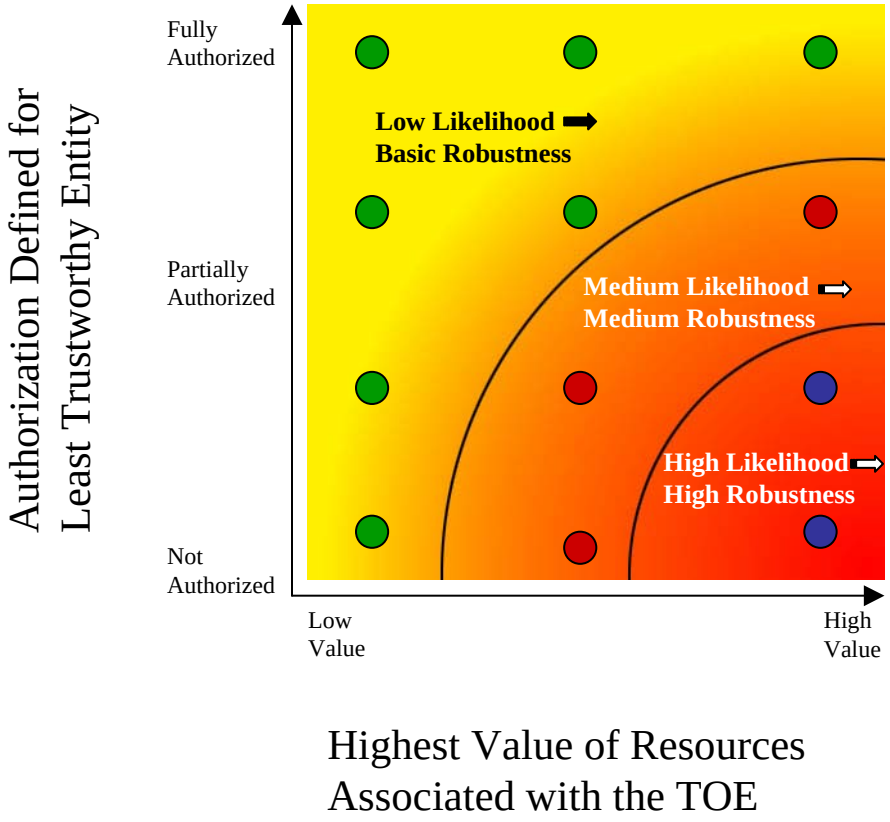

Figure 1-2 Likelihood of Attempted Compromise

29 The difficult part of this activity is differentiating the authorization of various entities, as well as determining the relative values of resources; (e.g., what constitutes "low value" data vs. "medium value" data). Because every organization will be different, a rigorous definition is not possible. In section 3 of this PP, the targeted threat level for a medium robustness TOE is characterized. This information is provided to help organizations using this PP ensure that the

functional requirements specified by this medium robustness PP are appropriate for their intended application of a compliant TOE.

1.3 Mutual Recognition of Common Criteria Certificates

30 The assurance requirements contained in this PP are equivalent to the Evaluated Assurance Level 4 (EAL 4) as defined in the "Common Criteria for Information Technology Security Evaluation Version 2.3" (CC) with augmentation. The augmented assurances are in the areas of vulnerability analysis/penetration testing, development, and covert channel analysis for cryptography. COTS operating systems meeting the requirements of this profile provide a medium level of robustness. Under the "Arrangement on the Mutual Recognition of Common Criteria Certificates in the field of Information Technology Security" document, only CC requirements at or below EAL 4 are mutually recognized. Because this profile exceeds the limits imposed by the "Arrangement on the Mutual Recognition of Common Criteria Certificates in the field of Information Technology Security" document, the US will recognize only certificates issued by the US evaluation scheme to meet this profile. Other national schemes are likewise under no obligation to recognize US certificates with assurance components exceeding EAL4.

1.4 Conventions

31 The notation, formatting, and conventions used in this protection profile (PP) are consistent with version 2.3 of the Common Criteria for Information Technology Security Evaluation. Font style and clarifying information conventions were developed to aid the reader.

32 The CC permits four functional component operations: assignment, iteration, refinement, and selection to be performed on functional requirements. These operations are defined in Common Criteria, Part 1, paragraph 6.4.1.3.2 as:

- assignment: allows the specification of an identified parameter;
- refinement: allows the addition of details or the narrowing of requirements;
- selection: allows the specification of one or more elements from a list; and
- iteration: allows a component to be used more than once with varying operations.

33 *Assignments or selections* left to be specified by the developer in subsequent security target documentation are italicized and identified between brackets ("[ ]"). In addition, when an assignment or selection has been left to the discretion of the developer, the text "assignment:" or "selection:" is indicated within the brackets. Assignments or selection created by the PP author (for the developer to complete) are bold, italicized, and between brackets ("[ ]"). CC selections completed by the PP author are underlined and CC assignments completed by the PP author are bold.

34 *Refinements* are identified with "**Refinement:**" right after the short name. They permit the addition of extra detail when the component is used. The underlying notion of a refinement is that of narrowing. There are two types of narrowing possible: narrowing of implementation and

narrowing of scope[2]. Additions to the CC text are specified in bold. Deletions of the CC text are identified in the "End Notes" with a bold number after the element ("**8**").

35 *Iterations* are identified with a number inside parentheses ("(#)"). These follow the short family name and allow components to be used more than once with varying operations.

36 *Explicit Requirements* are allowed to create requirements should the Common Criteria not offer suitable requirements to meet the PP needs. The naming convention for explicit requirements is the same as that used in the CC. To ensure these requirements are explicitly identified, the word "**Explicit:**" appears before the component behavior name to alert the reader. Additionally, the ending "_EXP" is appended to the newly created short name and the component and the element names are bolded. However, most of the explicit requirements are based on existing CC requirements.

37 *Application Notes* are used to provide the reader with additional requirement understanding or to clarify the author's intent. These are italicized and usually appear following the element needing clarification.

38 Table 1.1 provides examples of the conventions (explained in the above paragraphs) for the permitted operations.

Table 1.1 - Functional Requirements Operation Conventions

Convention	Purpose	Operation
Bold	The purpose of bolded text is used to alert the reader that additional text has been added to the CC. This could be an assignment that was completed by the PP author or a refinement to the CC statement. Examples: FAU_SAR.1.1 The TSF shall provide **authorized administrators** with the capability to read **all audit information** from the audit records. FTA_MCS.1.1 **Refinement:** The TSF shall restrict the maximum number of concurrent **interactive** sessions that belong to the same user.	(Completed) Assignment or Refinement

[2] US interpretation #0362: Scope of Permitted Refinements

Convention	Purpose	Operation
Italics	The purpose of italicized text is to inform the reader of an assignment or selection operation to be completed by the developer or ST author. It has been left as it appears in the CC requirement statement. Examples: FTA_SSL.1.1The TSF shall lock an interactive session after *[assignment: a time interval of user inactivity]* by: a) Clearing or overwriting display devices, making the current contents unreadable. b) Disabling any activity of the user's data access/display devices other than unlocking the session. FDP_RIP.2.1 **Refinement:** The TSF shall ensure that any previous information content of a resource is made unavailable upon the *[selection: allocation of the resource to, deallocation of the resource from]* all objects **other than those associated with cryptographic keys and critical cryptographic security parameters as described in FCS_CKM.4.1 and FCS_CKM_EXP.2.5**.	Assignment (to be completed by developer or ST author) or Selection (to be completed by developer or ST author)
<u>Underline</u>	The purpose of underlined text is to inform the reader that a choice was made from a list provided by the CC selection operation statement. Example: FAU_STG.1.2 The TSF shall be able to <u>prevent</u> modifications to the audit records.	Selection (completed by PP author)

Convention	Purpose	Operation
Bold & Italics	The purpose of bolded and italicized text is to inform the reader that the author has added new text to the requirement and that an additional vendor action needs to be taken. Example: FIA_UAU.1.1 **Refinement:** The TSF shall allow **read access to *[assignment: list of public objects]*** on behalf of the user to be performed before the user is authenticated. FCS_CKM.2.1 – The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method ***[selection: Manual (Physical) Method, Automated (Electronic Method), Manual Method and Automated Method]*** that meets the …	Assignment (added by the PP author for the developer or ST author to complete) or Selection (added by the PP author for the developer or ST author to complete)
Parentheses (Iteration #)	The purpose of using parentheses and an iteration number is to inform the reader that the author has selected a new field of assignments or selections with the same requirement and that the requirement will be used multiple times. Iterations are performed at the component level. The component behavior name includes information specific to the iteration between parentheses. Example: 5.5.3.1 Management of TSF Data (for general TSF data) (FMT_MTD.1(1)) FMT_MTD.1.1(1) The TSF shall restrict the ability to **create**, query, modify, delete, and clear the **security-relevant TSF data except for audit records, user security attributes, authentication data, and critical cryptographic security parameters** to **the authorized administrator**. 5.5.3.2 Management of TSF Data (for audit records) (FMT_MTD.1(2)) FMT_MTD.1.1(2) The TSF shall restrict the ability to query, delete, and clear the **audit records** to **authorized administrators**.	Iteration 1 (of component) Iteration 2 (of component)

Convention	Purpose	Operation
Explicit: (**_EXP**)	The purpose of using **Explicit:** before the family or component behavior name is to alert the reader and to explicitly identify a newly created component. To ensure these requirements are explicitly identified, the "_EXP" is appended to the newly created short name and the component and element names are bolded. Example: **5.5.7.1 Explicit: Internal TSF Data Consistency (FPT_TRC_EXP.1)** **FPT_TRC_EXP.1.1** The TSF shall ensure that TSF data is consistent between parts of the TOE by providing a mechanism to bring inconsistent TSF data into a consistent state in a timely manner.	Explicit Requirement
Endnotes	The purpose of endnotes is to alert the reader that the author has deleted Common Criteria text. An endnote number is inserted at the end of the requirement, and the endnote is recorded on the last page of the section. The endnote statement first states that a deletion was performed and then provides the rationale. Following is the family behavior or requirement in its original and modified form. A strikethrough is used to identify deleted text and bold for added text. A text deletion rationale is provided. Examples: Text as shown: FPT_TST.1.2 **Refinement:** The TSF shall provide authorized **administrators** with the capability to verify the integrity of TSF data.**18** Endnote statement: **18** A deletion of CC text was performed in FPT_TST.1.2. Rationale: The word "users" was deleted to replace it with the role of "authorized administrator". Only authorized administrators should be given the capability to verify the integrity of the TSF data. FPT_TST.1.2 **Refinement:** The TSF shall provide authorized ~~users~~ **administrators** with the capability to verify the integrity of TSF data.	Refinement

1.5 Glossary of Terms

39 This profile uses the terms described in this section to aid in the application of the requirements. The numbers specified between brackets ("[#]") at the end of some definitions point to the "References" section to identify where these definitions were obtained.

Access	Interaction between an entity and an object that results in the flow or modification of data [12].
Access control	Security service that controls the use of resources[3] and the disclosure and modification of data[4].
Accountability	Tracing each activity in an IT system to the entity responsible for the activity.
Administrator	An authorized user who has been specifically granted the authority to manage some portion or all of the TOE and thus whose actions may affect the TSP. Administrators may possess special privileges that provide capabilities to override portions of the TSP.
Assurance	A measure of confidence that the security features of an IT system are sufficient to enforce its security policy.
Asymmetric cryptographic system	A system involving two related transformations; one determined by a public key (the public transformation), and another determined by a private key (the private transformation) with the property that it is computationally infeasible to determine the private transformation (or the private key) from knowledge of the public transformation (and the public key).
Asymmetric key	The corresponding public/private key pair needed to determine the behavior of the public/private transformations that comprise an asymmetric cryptographic system.
Attack	An intentional act attempting to violate the security policy of an IT system.
Authentication	Security measure that verifies a claimed identity.
Authentication data	Information used to verify a claimed identity.
Authorization	Permission, granted by an entity authorized to do so, to perform functions and access data.
Authorized user	An authenticated user who may, in accordance with the TSP, perform an operation.

[3] hardware and software

[4] stored or communicated

Availability	Timely[5], reliable access to IT resources.
Compromise	Violation of a security policy.
Confidentiality	A security policy pertaining to disclosure of data.
Critical cryptographic security parameters	Security-related information (e.g., cryptographic keys, cryptographic seeds) appearing in plaintext or otherwise unprotected form and whose disclosure or modification can compromise the security of a cryptographic module or the security of the information protected by the module.
Cryptographic administrator	An authorized user who has been granted the authority to perform cryptographic initialization and management functions. These users are expected to use this authority only in the manner prescribed by the guidance given to them.
Cryptographic boundary	An explicitly defined contiguous perimeter that establishes the physical bounds (for hardware) or logical bounds (for software) of a cryptographic module.
Cryptographic key (key)	A parameter used in conjunction with a cryptographic algorithm that determines [8]: – the transformation of plaintext data into ciphertext data, – the transformation of ciphertext data into plaintext data, – a digital signature computed from data, – the verification of a digital signature computed from data, or – a data authentication code computed from data.
Cryptographic module	The set of hardware, software, and/or firmware that implements approved security functions (including cryptographic algorithms and key generation) and is contained within the cryptographic boundary.
Cryptographic module security policy	A precise specification of the security rules under which a cryptographic module must operate.
Defense-in-depth	A security design strategy whereby layers of protection are utilized to establish an adequate security posture for an IT system.
Discretionary Access Control (DAC)	A means of restricting access to objects based on the identity of subjects and groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject [12].

[5] according to a defined metric

Embedded cryptographic module	One that is built as an integral part of a larger and more general surrounding system (i.e., one that is not easily removable from the surrounding system).
Enclave	A collection of entities under the control of a single authority and having a homogeneous security policy. They may be logical, or based on physical location and proximity [2].
Entity	A subject, object, user or external IT device.
Identity	A means of uniquely identifying an authorized user of the TOE.
Named object	An object that exhibits all of the following characteristics: - The object may be used to transfer information between subjects of differing user identities within the TSF. - Subjects in the TOE must be able to request a specific instance of the object. - The name used to refer to a specific instance of the object must exist in a context that potentially allows subjects with different user identities to request the same instance of the object.
National Security Systems	Any telecommunications or information system operated by the United States Government, the function, operation, or use of which: (a) involves intelligence activities; (b) involves cryptologic activities related to national security; (c) involves command and control of military forces; (d) involves equipment that is an integral part of a weapon or weapon system; or (e) is critical to the direct fulfillment of military or intelligence missions and does not include a system that is to be used for routine administrative and business applications (including payroll, finance, logistics, and personnel management applications) [27].
Non-persistent key	A cryptographic key, such as a key used to encrypt or decrypt a single message or a session that is ephemeral in the system.
Object	An entity under the control of the TOE that contains or receives information and upon which subjects perform operations.
Operating environment	The total environment in which a TOE operates. It includes the physical facility and any physical, procedural, administrative and personnel controls [2].
Operational key	Key intended for protection of operational information or for the production or secure electrical transmissions of key streams.

Persistent key	A cryptographic key which must be maintained between sessions or processes. Generally, a key is persistent because the data it protects is persistent (e.g., an encrypted file) or because it is tied to a user (e.g. a user's private key). Contrast with a session key such as an IPsec key which protects data in transit.
Persistent storage	All types of data storage media that maintain data across system boots (e.g., hard disk, CD, DVD).
Public object	An object for which the TSF unconditionally permits all entities "read" access. Only the TSF or authorized administrators may create, delete, or modify the public objects.
Resource	A fundamental element in an IT system (e.g., processing time, disk space, and memory) that may be used to create the abstractions of subjects and objects.
Role	A unique set of TOE-defined functionality limited to a specific set of authorized users.
Secure State	Condition in which all TOE security policies are enforced.
Security attributes	TSF data associated with subjects, objects and users that is used for the enforcement of the TSP.
Security-enforcing	A term used to indicate that the entity (e.g., module, interface, subsystem) is related to the enforcement of the TOE security policies.
Security-supporting	A term used to indicate that the entity (e.g., module, interface, subsystem) is not security-enforcing however, its implementation must still preserve the security of the TSF.
Single-level system	A system that is used to process data of a single security level.
Split key	A variable that consists of two or more components that must be combined to form the operational key variable. The combining process excludes concatenation or interleaving of component variables.
Subject	An active entity within the TSC that causes operations to be performed. Subjects can come in two forms: trusted and untrusted. Trusted subjects are exempt from part or all of the TOE security policies. Untrusted subjects are bound by all TOE security policies.
Symmetric key	A single, secret key used for both encryption and decryption in symmetric cryptographic algorithms.

System High environment	An environment where all authorized users, with direct or indirect access, have all of the following: a) valid security clearances for all information within the environment, b) formal access approval and signed non-disclosure agreements for all the information stored and/or processed (including all compartments, sub-compartments and/or special access information), and c) valid need-to-know for some of the information contained within the environment.
Threat	Capabilities, intentions and attack methods of adversaries, or any circumstance or event, with the potential to violate the TOE security policy.
User	Any person who interacts with the TOE.
Vulnerability	A weakness that can be exploited to violate the TOE security policy.

1.6 Document Organization

40 *Section 1* provides the introductory material for the protection profile.

41 *Section 2* describes the Target of Evaluation in terms of its envisaged usage and connectivity.

42 *Section 3* defines the expected TOE security environment in terms of the threats to its security, the security assumptions made about its use, and the security policies that must be followed.

43 *Section 4* identifies the security objectives derived from the threats and policies.

44 *Section 5* identifies and defines the security functional requirements from the CC that must be met by the TOE in order for the functionality-based objectives to be met.

45 *Section 6* identifies the security assurance requirements.

46 *Section 7* provides a rationale to explicitly demonstrate that the information technology security objectives satisfy the policies and threats. Arguments are provided for the coverage of each policy and threat. The section then explains how the set of requirements are complete relative to the objectives, and that each security objective is addressed by one or more component requirements. Arguments are provided for the coverage of each objective.

47 *Section 8* identifies background material used as reference to create this profile.

48 *Appendix A* defines frequently used acronyms.

49 *Appendix B* lists cryptographic standards, policies, and other related publications that have been identified in section 5 of this protection profile.

50 *Appendix C* describes the statistical tests the must be performed to the random number generators.

51 *Appendix D* lists the randomizer qualification statistical test suite and describes the randomizer qualification test process.

52 *Appendix E* provides the rationale for the explicit Development Documentation (ADV) assurance requirements.

2. Target of Evaluation (TOE) Description

2.1 Product Type

53 This protection profile specifies requirements for general-purpose, multi-user, COTS operating systems together with the underlying hardware for use in National Security Systems. Such operating systems are typically employed in a networked office automation environment (see Figure 2.1) containing file systems, printing services, network services and data archival services and can host other applications (e.g., mail, databases). This profile does not specify any security characteristics of security-hardened devices (e.g. guards, firewalls) that provide environment protection at network boundaries. **When this TOE is used in composition with other products to make up a larger national security system, the boundary protection must provide the appropriate security mechanisms, cryptographic strengths and assurances as approved by NSA to ensure adequate protection for the security and integrity of this TOE and the information it protects.**

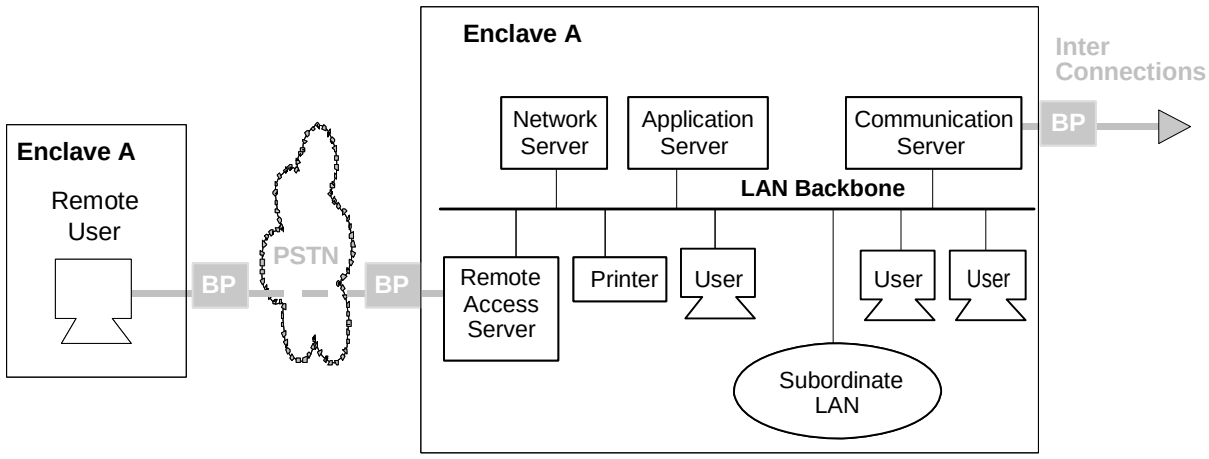

Figure 2-1 TOE Environment

2.2 General TOE Functionality

54 Conformant operating systems include the following security features:

– Identification and Authentication which mandates authorized users to be uniquely identified and authenticated before accessing information stored on the system;

- Discretionary Access Control (DAC) which restricts access to objects based on the identity of subjects and groups to which they belong, and allows authorized users to specify protection for objects that they control;
- Cryptographic services which provide mechanisms to protect TSF code and data and also provide support to allow authorized users and applications to encrypt, decrypt, hash, and digitally sign data as it resides within the system and as it is transmitted to other systems; and
- Audit services which allow authorized administrators to detect and analyze potential security violations.

55 Requirements not addressed in this PP include:

- mechanisms or services to ensure availability of data residing on the TOE.[6],
- mechanisms or services to ensure integrity of user data residing on the TOE, and
- complete physical protection mechanisms, which must be provided by the environment.

2.3 Cryptographic Requirements

56 The TOE cryptographic services must provide both a level of functionality and assurance regardless of its implementation (software, hardware, or any combination thereof). This is achieved by meeting both the NIST FIPS PUB 140-2 standard and all additional requirements as stated in this PP (refer to Appendix B for relevant cryptographic standards, policies, and other publications).

57 For cryptographic services fully implemented in hardware, all FIPS PUB 140-2 Level 3 requirements as well as all additional requirements identified in this PP, must be met. For all other implementations (i.e., software, or a combination of software and hardware), all the requirements identified in FIPS PUB 140-2 Security Level 1 plus some of the requirements for FIPS PUB 140-2 Security Level 3 (namely, those in the areas of: Cryptographic Module Ports and Interfaces; Roles, Services and Authentication; Cryptographic Key Management; Design Assurance; and Security Level 4 Self Tests as defined by this PP); and all additional requirements identified in this PP must be met. These two implementations, with the exception of the Electromagnetic Interference/Electromagnetic Compatibility requirements, are equivalent in intent and counter the identified threats in this protection profile.

58 For convenience, Section 5.2 of this PP identifies where a NIST certification is required and against what standard. To meet this PP, the vendor must have a NIST certification and receive NSA approval for compliance to Section 5.2 and all other crypto-related requirements in this PP.

2.4 TOE Operational Environment

59 The intended operational environment implements the DoD Defense-in-Depth strategy to allow the use of COTS products in National Security System environments. The fundamental strategy

[6] If availability requirements exist, the environment must provide the required mechanisms (e.g., mirrored/duplicated data).

is that layers of IA solutions are needed to establish an adequate IA posture. By implementing appropriate levels of protection in key areas in the system architecture, an effective set of safeguards can be tailored according to each organization's unique needs.

60 It is assumed that the TOE environment is under the control of a single administrative authority and has a homogeneous system security policy, including personnel and physical security. This environment can be specific to an organization or a mission and may also contain multiple networks or enclaves. Enclaves may be logical or be based on physical location and proximity.

61 The TOE may be accessible by external IT systems that are beyond the environment's security policies. The users of these external IT systems are similarly beyond the control of the operating system's policies. Although the users of these external systems are authorized in their environments, they are outside the scope of control of this particular environment so nothing can be presumed about their intent. They must be viewed as potentially hostile.

62 PP conformant systems are suitable for use in unclassified environments which process administrative, private, and sensitive/proprietary information and are candidates for classified environments that utilize appropriate systems engineering and defense-in-depth strategies. However, when an organization's most sensitive/proprietary information is to be sent from the TOE to another system across a publicly accessible network, the organization should also apply additional protection at the network boundaries.

3. TOE Security Environment

63 This section defines the expected TOE security environment in terms of the threats, security assumptions, and the security policies that must be followed for the medium robustness TOE.

3.1 Use of Medium Robustness

64 A medium robustness TOE is considered sufficient protection for environments where the likelihood of an attempted compromise is medium[7]. This implies that the motivation of the threat agents will be average in environments that are suitable for TOEs of medium robustness. Note that this also implies that the resources and expertise of the threat agents really are not factors that need to be considered, because highly sophisticated threat agents may not be motivated to use great expertise or extensive resources in an environment where medium robustness is suitable.

65 The medium motivation of the threat agents can be reflected in a variety of ways. One possibility is that the value of the data processed or protected by the TOE will be only medium, thus providing little motivation of even a totally unauthorized entity to attempt to compromise the data. Another possibility, (where higher value data is processed or protected by the TOE) is that the procuring organization will provide environmental controls (that is, controls that the TOE itself does not enforce) in order to ensure that threat agents that have generally high motivation levels (because of the value of the data) cannot logically or physically access the TOE (e.g., all users are "vetted" to help ensure their trustworthiness, and connectivity to the TOE is restricted).

3.2 Threat Agent Characterization

66 In addition to helping define the robustness appropriate for a given environment, the threat agent is a key component of the formal threat statements in the PP. Threat agents are typically characterized by a number of factors such as *expertise, available resources, and motivation*. Because each robustness level is associated with a variety of environments, there are corresponding varieties of specific threat agents (that is, the threat agents will have different combinations of motivation, expertise, and available resources) that are valid for a given level of robustness. The following discussion explores the impact of each of the threat agent factors on the ability of the TOE to protect itself (that is, the robustness required of the TOE).

67 The *motivation* of the threat agent seems to be the primary factor of the three characteristics of threat agents outlined above. Given the same expertise and set of resources, an attacker with low motivation may not be as likely to attempt to compromise the TOE. For example, an entity with

[7] An alternative perspective to thinking of the robustness level in terms of "likelihood of attempted compromise" is to consider the damage to the organization that would result if a TOE compromise were to occur. These two notions (likelihood of compromise and damage resulting from compromise) are parallel notions. They both are intrinsically linked to the value of the data being processed. The more valuable/sensitive the data, the greater the likelihood that an adversary will attempt to compromise the TOE, similarly the greater the damage to the organization that would result from such compromise.

no authorization to low value data nonetheless has low motivation to compromise the data because of its low value; thus a basic robustness TOE should offer sufficient protection. Likewise, fully authorized users with access to highly valued data similarly have low motivation to attempt to compromise the data because of their authorization, thus again a basic robustness TOE should be sufficient.

68 Unlike the motivation factor, however, the same can't be said for *expertise*. A threat agent with low motivation and low expertise is just as unlikely to attempt to compromise a TOE as an attacker with low motivation and high expertise; this is because the attacker with high expertise does not have the motivation to compromise the TOE even though they may have the expertise to do so. The same argument can be made for *resources* as well.

69 Therefore, when assessing the robustness needed for a TOE, the motivation of threat agents should be considered a "high water mark". *That is, the robustness of the TOE should increase as the motivation of the threat agents increases.*

70 Having said that, the relationship between expertise and resources is somewhat more complicated. In general, if resources include factors other than just raw processing power (money, for example), then expertise should be considered to be at the same "level" (low, medium, high, for example) as the resources because money can be used to purchase expertise. Expertise in some ways is different, because expertise in and of itself does not automatically procure resources. However, it may be plausible that someone with high expertise can procure the requisite amount of resources by virtue of that expertise.

71 It may not make sense to distinguish between these two factors; in general, it appears that the only effect these may have is to lower the robustness requirements. For instance, suppose an organization determines that, because of the value of the resources processed by the TOE and the trustworthiness of the entities that can access the TOE, the motivation of those entities would be "medium". This normally indicates that a medium robustness TOE would be required because the likelihood that those entities would attempt to compromise the TOE to get at those resources is in the "medium" range. However, now suppose the organization determines that the entities (threat agents) that are the least trustworthy have no resources and are unsophisticated. In this case, even though those threat agents have medium motivation, the likelihood that they would be able to mount a successful attack on the TOE would be low, and so a basic robustness TOE may be sufficient to counter that threat.

72 It should be clear from this discussion that there is no "cookbook" or mathematical answer to the question of how to specify exactly the level of motivation, the amount of resources, and the degree of expertise for a threat agent so that the robustness level of TOEs facing those threat agents can be rigorously determined. However, an organization can look at combinations of these factors and obtain a good understanding of the likelihood of a successful attack being attempted against the TOE. Each organization wishing to procure a TOE must look at the threat factors applicable to their environment; discuss the issues raised in the previous paragraph; consult with appropriate accreditation authorities for input; and document their decision regarding likely threat agents in their environment.

73 The important general points to make are:

- The motivation for the threat agent defines the upper bound with respect to the level of robustness required for the TOE
- A threat agent's expertise and/or resources that is "lower" than the threat agent's motivation (e.g., a threat agent with high motivation but little expertise and few resources) may lessen the robustness requirements for the TOE (see next point, however).
- The easy availability (e.g., via the Internet or "hacker chat rooms") of attacks that would normally require high expertise and/or high availability of resources introduces a problem when trying to define the expertise of, or resources available to, a threat agent.

3.3 Threats

74 The following threats are addressed by PP compliant TOEs:

T.ADMIN_ERROR	An administrator may incorrectly install or configure the TOE resulting in ineffective security mechanisms.
T.ADMIN_ROGUE	An authorized administrator's intentions may become malicious resulting in user or TSF data being compromised.
T.AUDIT_COMPROMISE	A malicious user or process may view audit records, cause audit records to be lost or modified, or prevent future audit records from being recorded, thus masking a user's action.
T.CRYPTO_COMPROMISE	A malicious user or process may cause key, data or executable code associated with the cryptographic functionality to be inappropriately accessed (viewed, modified, or deleted), thus compromising the cryptographic mechanisms and the data protected by those mechanisms.
T.EAVESDROP	A malicious user or process may observe or modify TSF data transmitted between physically separated parts of the TOE.
T.MASQUERADE	A malicious user, process, or external IT entity may masquerade as an authorized entity in order to gain unauthorized access to data or TOE resources.
T.POOR_DESIGN	Unintentional or intentional errors in requirements specification or design of the TOE may occur, leading to flaws that may be exploited by a malicious user or program.
T.POOR_IMPLEMENTATION	Unintentional or intentional errors in implementation of the TOE design may occur, leading to flaws that may be exploited by a malicious user or program.

T.POOR_TEST	Lack of or insufficient tests to demonstrate that all TOE security functions operate correctly may result in incorrect TOE behavior being undiscovered thereby causing potential security vulnerabilities.
T.REPLAY	A user may gain inappropriate access to the TOE by replaying authentication information, or may cause the TOE to be inappropriately configured by replaying TSF data or security attributes.
T.RESIDUAL_DATA	A user or process may gain unauthorized access to data through reallocation of TOE resources from one user or process to another.
T.RESOURCE_EXHAUSTION	A malicious process or user may block others from system resources (i.e., system memory, persistent storage, and processing time) via a resource exhaustion denial of service attack.
T.SPOOFING	A malicious user, process, or external IT entity may misrepresent itself as the TOE to obtain authentication data.
T.TSF_COMPROMISE	A malicious user or process may cause TSF data or executable code to be inappropriately accessed (viewed, modified or deleted).
T.UNATTENDED_SESSION	A user may gain unauthorized access to an unattended session.
T.UNAUTHORIZED_ACCESS	A user may gain unauthorized access (view, modify, delete) to user data.
T.UNIDENTIFIED_ACTIONS	The administrator may fail to notice potential security violations, thus preventing the administrator from taking action against a possible security violation.
T.UNKNOWN_STATE	When the TOE is initially started or restarted after a failure, the security state of the TOE may be unknown.

3.4 Security Policy

75 The following organizational security policies are addressed by PP compliant TOEs:

P.ACCESS_BANNER	The TOE shall display an initial banner describing restrictions of use, legal agreements, or any other appropriate information to which users consent by accessing the TOE.
P.ACCOUNTABILITY	The users of the TOE shall be held accountable for their actions within the TOE.

P.AUTHORIZATION	The TOE shall limit the extent of each user's abilities in accordance with the TSP.
P.AUTHORIZED_USERS	Only those users who have been authorized to access the information within the TOE may access the TOE.
P.CRYPTOGRAPHY	The TOE shall use NIST FIPS validated cryptography as a baseline with additional NSA-approved methods for key management (i.e., generation, access, distribution, destruction, handling, and storage of keys) and for cryptographic operations (i.e., encryption, decryption, signature, hashing, key exchange, and random number generation services).
P.I_AND_A	All users must be identified and authenticated prior to accessing any controlled resources with the exception of public objects.
P.INDEPENDENT_TESTING	The TOE must undergo independent testing as part of an independent vulnerability analysis.
P.NEED_TO_KNOW	The TOE must limit the access to data in protected resources to those authorized users who have a need to know that data.
P.RATINGS_MAINTENANCE	A plan for procedures to maintain the TOE's rating must be in place.
P.REMOTE_ADMIN_ACCESS	Any remote administration shall be securely managed by the TOE.
P.ROLES	The TOE shall provide multiple administrative roles for secure administration of the TOE. These roles shall be separate and distinct from each other.
P.SYSTEM_INTEGRITY	The TOE shall provide the ability to periodically validate its correct operation and, with the help of administrators, it must be able to recover from any errors that are detected.
P.TRACE	The TOE shall provide the ability to review the actions of individual users.
P.TRUSTED_RECOVERY	Procedures and/or mechanisms shall be provided to assure that, after a TOE failure or other discontinuity, recovery without a protection compromise is obtained.
P.VULNERABILITY_ANALYSIS_AND_TEST	The TOE must undergo a vulnerability analysis and penetration testing by NSA to demonstrate that the TOE is resistant to an attacker possessing a moderate attack potential.

3.5 Security Usage Assumptions

76 The specific conditions below are assumed to exist in a PP-compliant TOE environment:

A.PHYSICAL	It is assumed that the IT environment provides the TOE with appropriate physical security, commensurate with the value of the IT assets protected by the TOE.

4. Security Objectives

77 This section defines the security objectives for the TOE and its environment. These objectives are suitable to counter all identified threats and cover all identified organizational security policies and assumptions. The TOE security objectives are identified with "O." appended to the beginning of the name and the environment objectives are identified with "OE." appended to the beginning of the name.

4.1 TOE Security Objectives

O.ACCESS	The TOE will ensure that users gain only authorized access to it and to resources that it controls.
O.ACCESS_HISTORY	The TOE will display information (to authorized users) related to previous attempts to establish a session.
O.ADMIN_ROLE	The TOE will provide administrator roles to isolate administrative actions.
O.ADMIN_GUIDANCE	The TOE will provide administrators with the necessary information for secure management of the TOE.
O.AUDIT_GENERATION	The TOE will provide the capability to detect and create records of security relevant events associated with users.
O.AUDIT_PROTECTION	The TOE will provide the capability to protect audit information.
O.AUDIT_REVIEW	The TOE will provide the capability to selectively view audit information and alert the administrator of identified potential security violations.
O.CHANGE_MANAGEMENT	The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled throughout the TOE's development.
O.CORRECT_TSF_OPERATION	The TOE will provide a capability to test the TSF to ensure the correct operation of the TSF in its operational environment.
O.CRYPTOGRAPHIC_PROTECTION	The TOE will support separation of the cryptography from the rest of the TSF.
O.CRYPTOGRAPHIC_SERVICES	The TOE will make encryption services available to authorized users and/or user applications.

O.DISCRETIONARY_ACCESS	The TOE will control access to resources based upon the identity of users and groups of users.
O.DISCRETIONARY_USER_CONTROL	The TOE will allow authorized users to specify which resources may be accessed by which users and groups of users.
O.DISPLAY_BANNER	The TOE will display an advisory warning regarding use of the TOE.
O.ENCRYPTED_CHANNEL	Encryption will be used to provide confidentiality of TSF data in transit to remote parts of the TOE.
O.FUNCTIONAL_TESTING	The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.
O.INSTALL_GUIDANCE	The TOE will be delivered with the appropriate installation guidance to establish and maintain TOE security.
O.MANAGE	The TOE will provide all the functions and facilities necessary to support the authorized administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.
O.PENETRATION_TESTING	The TOE will undergo independent penetration testing to demonstrate that the design and implementation of the TOE prevents users from violating the TOE's security policies.
O.PROTECT	The TOE will provide mechanisms to protect user data and resources.
O.RATINGS_MAINTENANCE	Procedures to maintain the TOE's rating will be documented.
O.RECOVERY	Procedures and/or mechanisms will be provided to assure that recovery is obtained without a protection compromise, such as from system failure or discontinuity.
O.REPLAY_DETECTION	The TOE will provide a means to detect and reject the replay of authentication data, as well as, TSF data and security attributes.
O.RESIDUAL_INFORMATION	The TOE will ensure that any data contained in a protected resource is not available when the resource is reallocated.

O.RESOURCE_SHARING	The TOE shall provide mechanisms that mitigate user attempts to exhaust TOE resources (e.g., system memory, persistent storage, and processing time).
O.REFERENCE_MONITOR	The TOE will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure.
O.SECURE_STATE	The TOE will be able to verify the integrity of the TSF code and cryptographic data.
O.SOUND_DESIGN	The TOE will be designed using sound design principles and techniques. The TOE design, design principles and design techniques will be adequately and accurately documented.
O.SOUND_IMPLEMENTATION	The implementation of the TOE will be an accurate instantiation of its design.
O.TRAINED_USERS	The TOE will provide authorized users with the necessary guidance for secure use of the TOE, to include secure sharing of user data.
O.TRUSTED_PATH	The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data.
O.TSF_CRYPTOGRAPHIC_INTEGRITY	The TOE will provide cryptographic integrity mechanisms for TSF data while in transit to remote parts of the TOE.
O.USER_AUTHENTICATION	The TOE will verify the claimed identity of users.
O.USER_IDENTIFICATION	The TOE will uniquely identify users.
O.VULNERABILITY_ANALYSIS	The TOE will undergo appropriate vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with moderate attack potential to violate the TOE's security policies.

4.2 Environment Security Objectives

OE.PHYSICAL	Physical security will be provided for the TOE by the IT environment, commensurate with the value of the IT assets protected by the TOE.

5. Security Functional Requirements

78 This section contains detailed security functional requirements for the operating systems' trusted security functions (TSF) supporting single-level systems in medium robustness environments. The requirements are applied against the operating system in conjunction with the underlying hardware that supports it. The requirements contained in this section are either selected from Part 2 of the CC or have been explicitly stated (with short names in bold and ending in "_EXP"). Table 5.1 lists the explicit functional requirements in this section.

79 The cryptographic module plays an important role in the enforcement of the TOE security policies. For this reason, the cryptographic related requirements contain more detail than other requirements, in terms of refinements, iterations, and explicitly stated requirements. Refer to section 1.3 to see the notation and formatting used in this profile.

Table 5.1 - Explicit Functional Requirements

Explicit Component	Component Behavior Name
FCS_BCM_EXP.1	Baseline Cryptographic Module
FCS_CKM_EXP.1	Key Validation and Packaging
FCS_CKM_EXP.2	Cryptographic Key Handling and Storage
FCS_COA_EXP.1	Cryptographic Operations Availability
FCS_COP_EXP.1	Random Number Generation
FPT_TRC_EXP.1	Internal TSF Data Consistency
FPT_TST_EXP.1	TSF Testing
FTP_TRP_EXP.1	Trusted Path

5.1 Security Audit (FAU)

5.1.1 Security Audit Automatic Response (FAU_ARP)

5.1.1.1 Security Alarms (FAU_ARP.1)

FAU_ARP.1.1 **Refinement:** Upon detection of a potential security violation, the TSF shall **generate a warning message to the authorized administrator that requires explicit acknowledgement by the administrator**.1

Application Note: "Potential security violation" is an activity that, if continued unchecked, would lead to a security violation (e.g. repeated failed authentication attempts).

5.1.2 Security Audit Data Generation (FAU_GEN)

5.1.2.1 Audit Data Generation (FAU_GEN.1)

FAU_GEN.1.1 **Refinement:** The TSF shall be able to generate an audit record of the following auditable events:

a) Start-up and shutdown of the audit functions;

b) Start-up and shutdown of the TOE;

c) Uses of special permissions that circumvent the access control policies;

Application Note: These special permissions are typically those often used by authorized administrators.

d) All auditable events **listed in Table 5.2**; **and**

e) All other security relevant auditable events for the minimal level of audit.

Application Note: For other security relevant functions that are not included in this PP, the ST author defines a minimal level of audit.

Table 5.2 - Auditable Events[8]

Requirement	Audit events prompted by requirement
Security Alarms (FAU_ARP.1)	• Actions taken to address potential security violations.
Audit Data Generation (FAU_GEN.1)	(none)
User Identity Association (FAU_GEN.2)	(none)
Potential Violation Analysis (FAU_SAA.1)	• Enabling and disabling of any of the analysis mechanisms. • Automated responses provided by the security audit analysis mechanism.
Audit Review (FAU_SAR.1)	• Opening the audit records.
Restricted Audit Review (FAU_SAR.2)	• Unsuccessful attempts to read information from the audit records.
Selectable Audit Review (FAU_SAR.3)	(none)
Selective Audit (FAU_SEL.1)	• All modifications to the audit configuration that occur while the audit collection functions are operating.
Protected Audit Trail Storage (FAU_STG.1)	(none)
Prevention of Audit Data Loss (FAU_STG.4)	• Actions taken due to the audit storage failure.

[8] Not all listed events must be captured in separate audit records but the capability must exist to query the audit data based on any individual event.

Explicit: Baseline Cryptographic Module (FCS_BCM_EXP.1)	(none)
Cryptographic Key Generation (for symmetric keys) (FCS_CKM.1(1))	• Failure of the symmetric key generation process[9].
Cryptographic Key Generation (for asymmetric keys) (FCS_CKM.1(2))	• Failure of the asymmetric key generation process[9].
Cryptographic Key Distribution (FCS_CKM.2)	• Failure to properly complete the key distribution process[9].
Cryptographic Key Destruction (FCS_CKM.4)	• Failure of the key zeroization process[9].
Explicit: Cryptographic Key Validation and Packaging (FCS_CKM_EXP.1)	• Failure of a key validation technique[9].
Explicit: Cryptographic Key Handling and Storage (FCS_CKM_EXP.2)	• Failure in key handling or storage[9].
Cryptographic Operations Availability (FCS_COA_EXP.1)	(none)
Cryptographic Operation (for data encryption/decryption) (FCS_COP.1(1))	• Failure in encryption or decryption[9].
Cryptographic Operation (for cryptographic signature) (FCS_COP.1(2))	• Failure in cryptographic signature[9].
Cryptographic Operation (for cryptographic hashing) (FCS_COP.1(3))	• Failure in hashing function[9].
Cryptographic Operation (for cryptographic key agreement) (FCS_COP.1(4))	• Failure in cryptographic key exchange[9].
Explicit: Random Number Generation (FCS_COP_EXP.1)	• Failure in the randomization process[9].

[9] Typically, upon detection of a crypto-related failure, a system indication should be generated, and the system should transition to a known safe (secure) state. The generation of an audit log can provide a mechanism for capturing more information about a failed event. The exact content of the crypto-related audit log is implementation-dependent. However, the log should include information that could help pinpoint the part of the crypto-related process that failed, but without compromising the value of any critical cryptographic security parameters. In addition, the audit record requirements specified in FAU_GEN.1.2 should be considered and included where appropriate. As a simple example, detection of a key checkword error during an internal transfer of key might be implemented as follows: Generate a “Bad Key” error message to the system, prevent use of the bad key and zeroize it, and generate an audit record that includes the date of the event, the time of the event, “key checkword error”, bad key ID tag or subject/user associated with the bad key, and “failed key transfer during internal handling”.

Complete Access Control (FDP_ACC.2)	(none)
Security Attribute Based Access Control (FDP_ACF.1)	• All requests to perform an operation on an object covered by the SFP.
Full Residual Information Protection (FDP_RIP.2)	(none)
Authentication Failure Handling (FIA_AFL.1)	• The reaching of the threshold for the unsuccessful authentication attempts and the actions (e.g. disabling of a terminal) taken and the subsequent, if appropriate, restoration to the normal state (e.g. re-enabling of a terminal).
User Attribute Definition (FIA_ATD.1)	(none)
Verification of Secrets (FIA_SOS.1)	• Rejection or acceptance by the TSF of any tested secret.
Timing of Authentication (FIA_UAU.1)	• All use of the authentication mechanism.
Re-authenticating (FIA_UAU.6)	• All re-authentication attempts.
Protected Authentication Feedback (FIA_UAU.7)	(none)
Timing of Identification (FIA_UID.1)	• All use of the user identification mechanism, including the user identity provided.
User-Subject Binding (FIA_USB.1)	• Success and failure of binding of user security attributes to a subject (e.g. success and failure to create of a subject).
Management of Security Functions Behavior (for specification of auditable events) (FMT_MOF.1(1))	• All modifications in the behavior of the functions in the TSF.
Management of Security Functions Behavior (for authentication data) (FMT_MOF.1(2))	• All modifications in the behavior of the functions in the TSF.
Management of Security Attributes (FMT_MSA.1)	• All modifications of the values of security attributes.
Management of Security Attributes (for Object Ownership) (FMT_MSA.1(2))	• All modifications of the values of security attributes.
Secure Security Attributes (FMT_MSA.2)	• All offered and rejected values for a security attribute.
Static Attributes Initialization (FMT_MSA.3)	• Modifications of the default setting of permissive or restrictive rules. • All modifications of the initial values of security attributes.
Management of TSF Data (for general TSF data) (FMT_MTD.1(1))	• All modifications of the values of TSF data.
Management of TSF Data (for audit data) (FMT_MTD.1(2))	• All modifications of the values of audit data.

Management of TSF Data (for initialization of user security attributes) (FMT_MTD.1(3))	• All initializations of the values of user security attributes.
Management of TSF Data (for modification of user security attributes, other than authentication data) (FMT_MTD.1(4))	• All modifications of the values of user security attributes.
Management of TSF Data (for modification of authentication data) (FMT_MTD.1(5))	• All actions associated with modifications of the values of authentication data.
Management of TSF Data (for reading of authentication data) (FMT_MTD.1(6))	(none)
Management of TSF Data (for critical cryptographic security parameters) (FMT_MTD.1(7))	• All actions associated with modifications of the values of critical cryptographic security parameters.
Revocation (to authorized administrators) (FMT_REV.1(1))	• All attempts to revoke security attributes.
Revocation (to owners and authorized administrators) (FMT_REV.1(2))	• All attempts to revoke security attributes.
Time-Limited Authorization (FMT_SAE.1)	• Specification of the expiration time for an attribute. • Action taken due to attribute expiration.
Security Roles (FMT_SMR.2)	• Modifications to the group of users that are part of a role.
Assuming Roles (FMT_SMR.3)	• Explicit requests to assume a role. • Use of any function restricted to an authorized administrator role (identified in FMT_SMR.2).
Basic Internal TSF Data Transfer Protection (FPT_ITT.1)	(none)
TSF Data Integrity Monitoring (FPT_ITT.3)	• Detection of modification of TSF data.
Manual Recovery (FPT_RCV.1)	• The fact that a failure or service discontinuity occurred. • Resumption of the regular operation. • Type of failure or service discontinuity
Replay Detection (FPT_RPL.1)	• Detected replay
Non-Bypassability of the TSF (FPT_RVM.1)	(none)
SFP Domain Separation (FPT_SEP.2)	(none)
Reliable Time Stamps (FPT_STM.1)	• Changes to the time.

Internal TSF Data Consistency (FPT_TRC_EXP.1)	• Any detection of inconsistency between TSF data.
TSF Testing (FPT_TST_EXP.1)	• Execution of the TSF self tests and the results of the tests.
TSF Testing (for cryptography) (FPT_TST.1(1))	• Execution of the cryptography self tests and the results of the tests.
TSF Testing (for key generation components) (FPT_TST.1(2))	• Execution of the key generation component self tests and the results of the tests.
Maximum Quotas (for persistent storage) (FRU_RSA.1(1))	• Rejection of allocation operation due to persistent storage limits.
Maximum Quotas (for system memory) (FRU_RSA.1(2))	(none)
Maximum Quotas (for processing time) (FRU_RSA.1(3))	(none)
Limitation on scope of selectable attributes (FTA_LSA.1)	• All attempts at selecting a session security attribute.
Basic limitation on multiple concurrent sessions (FTA_MCS.1)	• Rejection of a new session based on the limitation of multiple concurrent sessions.
TSF-Initiated Session Locking (FTA_SSL.1)	• Locking of an interactive session by the session locking mechanism. • Any attempts at unlocking of an interactive session.
User-Initiated Locking (FTA_SSL.2)	• Locking of an interactive session by the session locking mechanism. • Any attempts at unlocking of an interactive session.
Default TOE Access Banners (FTA_TAB.1)	(none)
TOE Access History (FTA_TAH.1)	(none)
TOE Session Establishment (FTA_TSE.1)	• All attempts at establishment of a user session.
Trusted Path (FTP_TRP_EXP.1)	• All attempted uses of the trusted path functions. • Identification of the user associated with all trusted path failures, if available.

FAU_GEN.1.2 The TSF shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event; and

Application Note: "Subject identity" means user identity associated with the subject.

Application Note: For alarms, type of event refers to the cause of what triggered the alarm (not merely the fact that an alarm was triggered).

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST,

• the name of the object;

• for changes to TSF data (except for authentication data and critical cryptographic security parameters), the new and old values of the data;

• for cryptographic key failures, object attribute(s) and object value(s) that were involved in the failed operation excluding any sensitive information (e.g. secret or private keys);

• for cryptographic operation failures, applicable cryptographic mode(s) of operation, subject attributes and object attributes, excluding any sensitive information (e.g. secret or private keys);

• for authentication attempts, the origin of the attempt (e.g., terminal identifier);

• for assuming a role, the type of role, and the location of the request;

Application Note: TSF data includes access control attributes, user security attributes, definition of roles, and user authorizations.

Application Note: Other audit relevant information associated with security-relevant functions not included in this PP should be included within the audit records.

Application Note: "Location" refers to what ever means the TOE used to identify the point of entry when the interactive user session was established. The adequacy of this means is determined by other requirements (e.g., FPT_SEP, AVA_VLA).

5.1.2.2 User Identity Association (FAU_GEN.2)

FAU_GEN.2.1 The TSF shall be able to associate each auditable event with the identity of the user that caused the event.

Application Note: For failed login attempts no user identity association is required because the user is not under TSF control until after a successful identification/authentication.

5.1.3 Security Audit Analysis (FAU_SAA)

5.1.3.1 Potential Violation Analysis (FAU_SAA.1)

FAU_SAA.1.1 The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the enforcement of the SFRs.

FAU_SAA.1.2 The TSF shall enforce the following rules for monitoring audited events:

a) Accumulation or combination of **the following events** known to indicate a potential security violation:

1. **an administrator specified number of individual user authentication failures within an administrator specified time period,**
2. **an administrator specified number of Discretionary Access Control policy violation attempts by an individual user within an administrator specified time period,**

3. **any failure of the cryptographic self-tests,**
4. **any failure of the TSF self-tests,**
5. ***[assignment:* additional events from the set *of defined auditable events].***

b) [assignment: *any other rules*].

5.1.4 Security Audit Review (FAU_SAR)

5.1.4.1 Audit Review (FAU_SAR.1)

FAU_SAR.1.1 The TSF shall provide **authorized administrators** with the capability to read **all audit information** from the audit records.

Application Note: For a distributed system, the authorized administrator should be able to read all audit information within the TOE.

FAU_SAR.1.2 **Refinement:** The TSF shall provide the audit records in a manner suitable for the **authorized administrator** to interpret the information **using a tool to access the audit records.2**

Application Note: The tool provides a means to easily and efficiently review the audit records. It is expected (yet not necessary) that the tool satisfying this requirement will also satisfy the FAU_SAR.3 and FAU_SEL.1 requirements.

5.1.4.2 Restricted Audit Review (FAU_SAR.2)

FAU_SAR.2.1 The TSF shall prohibit all users read access to the audit records, except those users that have been granted explicit read-access.

5.1.4.3 Selectable Audit Review (FAU_SAR.3)

FAU_SAR.3.1 The TSF shall provide the ability to perform searches and sorting of audit data based on **the following attributes:**

a) user identity,

b) object identity,

c) date of the event,

d) time of the event,

e) type of event,

f) success of auditable security events, and

g) failure of auditable security events.

5.1.5 Security Audit Event Selection (FAU_SEL)

5.1.5.1 Selective Audit (FAU_SEL.1)

FAU_SEL.1.1 The TSF shall be able to include or exclude auditable events from the set of audited events based on the following attributes:

a) object identity,

b) user identity,

c) host identity,

d) event type,

e) success of auditable security events, and

f) failure of auditable security events.

5.1.6 Security Audit Event Storage (FAU_STG)

5.1.6.1 Protected Audit Trail Storage (FAU_STG.1)

FAU_STG.1.1 The TSF shall protect the stored audit records from unauthorized deletion.

FAU_STG.1.2 **Refinement:** The TSF shall be able to prevent modifications to the audit records.**3**

Application Note: In order to reduce the performance impact of audit generation, audit records are often temporarily buffered in memory before being written to the disk. In such implementations, these buffered records will be lost if the operation of the TOE is interrupted by hardware or power failures. The developer should document the expected loss in such circumstances and show that it has been minimized.

5.1.6.2 Prevention of Audit Data Loss (FAU_STG.4)

FAU_STG.4.1 **Refinement: When the audit trail becomes full,** the TSF shall **provide the authorized administrator the capability to** prevent auditable events, except those taken by the authorized **administrator (in the context of performing TOE maintenance)** and **generate an alarm to the authorized administrator**.**4**

5.2 Cryptographic Support (FCS)[10]

Application Note: All requirements specified in section 5.2 will be evaluated by NSA.

5.2.1 Explicit: Baseline Cryptographic Module (FCS_BCM_EXP)

5.2.1.1 Explicit: Baseline Cryptographic Module (FCS_BCM_EXP.1)

FCS_BCM_EXP.1.1 All cryptographic modules shall comply with FIPS PUB 140-2, "Security Requirements for Cryptographic Modules" when performing NIST-approved cryptographic functions in NIST-approved cryptographic modes of operation.

Application Note: A NIST FIPS 140-2 certification is required for all cryptographic functions in this PP that are covered in FIPS 140-2.

Application Note: FIPS PUB 140-2 is currently undergoing a regular five year review. In the near future, FIPS PUB 140-3 will supersede it.

FCS_BCM_EXP.1.2 Cryptographic functions and cryptographic modes of operation as identified in this Protection Profile shall be NIST-approved.

Application Note: In time, cryptographic requirements are expected to evolve such that cryptographic modules shall only contain cryptographic functions, cryptographic modes of operation, and other types of cryptographic processing that are compliant with public standards.

FCS_BCM_EXP.1.3 All cryptographic modules implemented in the TSF ***[selection:***

(1) Entirely in hardware shall have a minimum overall rating of FIPS PUB 140-2, Level 3,

(2) Entirely in software shall have a minimum overall rating of FIPS PUB 140-2, Level 1 and also meet FIPS PUB 140-2, Level 3 for the following: Cryptographic Module Ports and Interfaces; Roles, Services and Authentication; Cryptographic Key Management; and Design Assurance,

[10] In drafting specific requirements for this section for general-purpose operating systems, experts were consulted and their input was incorporated. The result is a very minimal set of crypto-related requirements chosen to be consistent with the other requirements of this Protection Profile. These crypto requirements are expected to be achievable in commercial products in the near term, and to gradually mature over time.

Evolving public standards on cryptographic functions and related areas have required the following interim approach to writing these cryptographic requirements for general purpose operating systems. This approach uses a variety of footnotes and application notes in an attempt to fill gaps, forewarn of future plans, and/or qualify interpretation of the existing referenced standards (sometimes specific draft versions). As a result, in many instances the presentation of the crypto requirements here is more cumbersome than desired. Still, today these requirements represent a step in the direction of helping to improve the security in COTS products. Over time, the approach and presentation will be expanded upon and refined. Correspondingly, the Protection Profile will be updated as the underlying public standards and the body of related special publications mature.

(3) As a combination of hardware and software shall have a minimum overall rating of FIPS PUB 140-2, Level 1 and also meet FIPS PUB 140-2, Level 3 for the following: Cryptographic Module Ports and Interfaces; Roles, Services and Authentication; and Cryptographic Key Management; Design Assurance.]

Application Note: "Combination of hardware and software" means that some part of the cryptographic functionality will be implemented as a software component of the TSF. The combination of a cryptographic hardware module and a software device driver whose sole purpose is to communicate with the hardware module is considered a hardware module rather than a "combination of hardware and software".

Application Note: The statistical RNG tests required for FIPS 140-2 Self Tests Security Level 3/4 are included in Appendix C of this protection profile..

5.2.2 Cryptographic Key Management (FCS_CKM)

5.2.2.1 Cryptographic Key Generation (for symmetric keys) (FCS_CKM.1(1))

FCS_CKM.1.1(1) **Refinement:** The TSF shall generate[11] **symmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm **as follows**: ***[selection:***

(1) a hardware random number generator (RNG) as specified in FCS_COP_EXP.1, and/or

(2) a software RNG as specified in FCS_COP_EXP.1, and/or

(3) a key establishment scheme as specified in FCS_COP.1(4) based upon public key cryptography using a software RNG as specified in FCS_COP_EXP.1, and/or a hardware RNG as specified in FCS_COP_EXP.1]

that meets the following: **5**

a) All cases: (i.e., any of the above)

- **NIST Special Publication 800-57, "Recommendation for Key Management"**

b) Case: Finite field-based key establishment schemes[12]

- **NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"**

Application Note: Any pseudorandom RNG used in these schemes for generating private values need to be seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this Protection Profile).

[11] This requirement applies strictly to **generation** of symmetric keys. **Validation** techniques for generated symmetric keys are discussed in FCS_CKM_EXP.1.1.

[12] For example, "classic" Diffie-Hellman-based schemes.

c) Case: RSA-based key establishment schemes

- **ANSI X9.31-1998 (May 1998), "Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)" for the generation of the RSA parameters**

Application Note: Although ANSI X9.31 is a standard intended for digital signatures, it is being used here for its coverage of the generation of RSA parameters since NIST Special Publication 800-56B and ANSI X9.44 are still under development.

Application Note: A pseudorandom RNG seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this Protection Profile) is used in the generation of these primes (RSA parameters).

d) Case: Elliptic curve-based key establishment schemes

- **NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"**
- **Only the "NIST curves" P-256, P-384 and P-521 (as defined in FIPS PUB 186-3, "Digital Signature Standard") may be used.**

Application Note: Any pseudorandom RNG used in these schemes for generating private values shall be seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this Protection Profile).

5.2.2.2 Cryptographic Key Generation (for asymmetric keys) (FCS_CKM.1(2))

FCS_CKM.1.1(2) **Refinement:** The TSF shall generate[13] **asymmetric**[14] cryptographic keys in accordance with a **domain parameter generator** and ***[selection:***

(1) a random number generator, and/or

(2) a prime number generator]

that meet the following: **6**

a) All cases: (i.e., any of the above)

- **NIST Special Publication 800-57, "Recommendation for Key Management"**
- **ANSI X9.80 (3 January 2000), "Prime Number Generation, Primality Testing, and Primality Certificates" using random integers with deterministic tests, or constructive generation methods**
- **Generated key strength shall be equivalent to, or greater than, a symmetric key strength of 128 bits using conservative estimates.**

[13] This requirement applies strictly to **generation** of asymmetric keys. **Validation** techniques for generated asymmetric keys are discussed in FCS_CKM_EXP.1.2.

[14] These are the keys/parameters (e.g., the public/private key pairs) underlying a public key-based key establishment scheme, not the session keys established by such schemes.

Application Note: The generated key strength of 2048-bit DSA and rDSA 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.

b) Case: For domain parameters used in finite field-based key establishment schemes[15]

- **NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"**

Application Note: Any pseudorandom RNG used in these schemes for generating private values are seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this Protection Profile).

c) Case: For domain parameters used in RSA-based key establishment schemes

- **ANSI X9.31-1998, "Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)" for the generation of the RSA parameters**

Application Note: Although ANSI X9.31 is a standard intended for digital signatures, it is being used here for its coverage of the generation of RSA parameters since NIST Special Publication 800-56B and ANSI X9.44 are still under development.

Application Note: A pseudorandom RNG seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this Protection Profile) is used in the generation of these primes (RSA parameters).

d) Case: For domain parameters used in elliptic curve-based key establishment schemes

- **NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"**
- **Only the "NIST curves" P-256, P-384 and P-521 (as defined in FIPS PUB 186-3, "Digital Signature Standard") may be used.**

Application Note: Any pseudorandom RNG used in these schemes for generating private values is seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this Protection Profile).

5.2.2.3 Cryptographic Key Distribution[16] (FCS_CKM.2)

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method ***[selection:***

(1) Manual (Physical) Method,

[15] For example, "classic" Diffie-Hellman-based scheme.

[16] Key Distribution (and key establishment) is typically addressed in terms of key transport methods or key agreement methods. Key transport methods are discussed in this section. Key agreement methods are addressed in FCS_COP.1(4) (Cryptographic Operation (for cryptographic key agreement)).

(2) Automated (Electronic) Method,

(3) Manual Method and Automated Method]

that meets the following:

a) Case: Manual (Physical) Methods:

- **The TSF shall support manual distribution of symmetric key in accordance with NIST Special Publication 800-57 "Recommendation for Key Management," and NIST Special Publication 800-56A "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"**

Application Note NIST Special Publication 800-56A "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" is only applicable when public key schemes are used in key transport methods.

- **The TSF shall support manual distribution of <u>public</u> asymmetric key material (certificates and/or keys) in accordance with DOD PKI for public key distribution using a certificate scheme for protection of public keys that meets the following:**
  1) **DoD X.509 Certificate Policy**
  2) **PKCS#12 v1.0, "Personal Information Exchange Syntax."**
- **The TSF shall support manual distribution of <u>private</u> asymmetric key material (certificates and/or keys) in accordance with DOD PKI for public key distribution using a certificate scheme with hardware tokens for protection of private keys that meets the following:**
  1) **DoD X.509 Certificate Policy**
  2) **PKCS #8 v1.2, "Private-Key Information Syntax Standard"**
  3) **PKCS #12 v1.0, "Personal Information Exchange Syntax"**
  4) **PKCS #5 v2.0, "Password-Based Encryption Standard, 25 Mar 1999 – Final"**
  5) **PKCS #11 v2.11, "Cryptographic Token Interface Standard."**

b) Case: Automated (Electronic) Methods:

- **The TSF shall automatically distribute symmetric keys in accordance with NIST Special Publication 800-57 "Recommendation for Key Management," and NIST Special Publication 800-56A "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography".**

Application Note: NIST Special Publication 800-56A "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" is only valid when public key schemes are used in key transport methods.

- **The TSF shall automatically distribute <u>public</u> asymmetric key material (certificates and/or keys) in accordance with DoD PKI for public key**

distribution using certificate schemes for the protection of public keys that meet the following:

1) **DoD X.509 Certificate Policy**
2) **PKCS#12 v1.0, "Personal Information Exchange Syntax)."**

- **The TSF shall support only manual distribution of private asymmetric key material (certificates and/or keys) in accordance with DOD PKI for public key distribution using certificate schemes with hardware tokens for protection of private keys that meet the following:**

1) **DoD X.509 Certificate Policy**
2) **PKCS #8 v1.2, "Private-Key Information Syntax Standard"**
3) **PKCS #12 v1.0, "Personal Information Exchange Syntax Standard"**
4) **PKCS #5 v2.0, "Password-Based Encryption Standard, 25 Mar 99—Final"**
5) **PKCS #11 v2.11, "Cryptographic Token Interface Standard."**

Application Note: DoD applications require Class 5 PKI to address worst case environments, but currently this class is just a concept. In the interim, a certificate scheme with hardware tokens for protection of private keys is approved under the added requirement that stronger protection mechanisms must be applied at the boundaries of the protected environment as stated earlier in this Protection Profile. When Class 5 certificates are fully established, they will be required.

5.2.2.4 Cryptographic Key Destruction (FCS_CKM.4)

FCS_CKM.4.1: **Refinement:** The TSF shall destroy cryptographic keys in accordance with a **cryptographic key destruction method** that meets the following: **7**

a) **FIPS PUB 140-2, "Security Requirements for Cryptographic Modules"**

b) **Zeroization of all plaintext cryptographic keys and all other critical cryptographic security parameters shall be immediate and complete.**

Application Note: The term "immediate" here is meant to impart some urgency to the destruction: it should happen as soon as practical after the key is no longer required to be in plaintext. It is certainly permissible to complete a critical section of code before destroying the key. However, the destruction shouldn't wait for idle time, and there shouldn't be any non-determined event (such as waiting for user input) which occurs before it is destroyed.

c) **For non-volatile memories other than EEPROM and Flash, the zeroization shall be executed by overwriting the key/critical cryptographic security parameter storage area three or more times using a different alternating data pattern each time.**

Application Note: Although verification of this zeroization of a plaintext key/critical cryptographic security parameter is desired here (by checking for the final known alternating data pattern), it is not required at this time. However, vendors are highly encouraged to incorporate this verification whenever possible into their implementations.

d) For volatile memory and non-volatile EEPROM and Flash memories, the zeroization shall be executed by overwriting the key/critical cryptographic security parameter storage area with a single direct overwrite consisting of a pseudo random pattern, followed by a read-verify.

Application Note: Zeroization of any storage, such as memory buffers, that is included in the path of a plaintext key/critical cryptographic security parameter is addressed in FCS_CKM_EXP.2 (Cryptographic Key Handling and Storage).

5.2.2.5 Explicit: Cryptographic Key Validation and Packaging (FCS_CKM_EXP.1)

FCS_CKM_EXP.1.1: The TSF shall apply validation techniques (e.g., parity bits or checkwords) to generated **symmetric** keys in accordance with NIST Special Publication 800-57, "Recommendation for Key Management."

FCS_CKM_EXP.1.2: The TSF shall apply validation techniques to generated **asymmetric** keys in accordance with NIST Special Publication 800-57, "Recommendation for Key Management" and the standards corresponding to the generation technique as called out in FCS_CKM.1.1(2).

FCS_CKM_EXP.1.3: Any public key certificates generated by the TSF shall be in accordance with DoD PKI certificate schemes.

Application Note: DoD applications require Class 5 PKI to address worst case environments, but currently this class is just a concept. In the interim, a certificate scheme with hardware tokens for protection of private keys is approved under the added requirement that stronger protection mechanisms must be applied at the boundaries of the protected environment as stated earlier in this Protection Profile. When Class 5 certificates are fully established, they will be required.

5.2.2.6 Explicit: Cryptographic Key Handling and Storage (FCS_CKM_EXP.2)

Application Note: NIST Special Publication 800-57, "Recommendation for Key Management" contains additional protection mechanisms that vendors are encouraged to implement. This document should be used as guidance for the key handling and storage requirements.

FCS_CKM_EXP.2.1: The TSF shall perform key entry and output in accordance with FIPS PUB 140-2, Level 3.

FCS_CKM_EXP.2.2: The TSF shall provide a means to ensure that keys are associated with the correct entities (i.e., person, group, or process) to which the keys are assigned.

FCS_CKM_EXP.2.3: The TSF shall perform a key error detection check on each transfer of key (internal, intermediate transfers).

Application Note: A parity check is an example of a key error detection check.

FCS_CKM_EXP.2.4: The TSF shall encrypt or split persistent secret and private keys when not in use.

Application Note: A persistent key, such as a file encryption key, is one that must be available in the system over long periods of time. A non-persistent key, such as a key used to encrypt or decrypt a single message or a session, is one that is ephemeral in the system.

Application Note: "When not in use" is interpreted in the strictest sense so that persistent keys only exist in plaintext form during intervals of operational necessity. For example, a file encryption key exists in plaintext form only during actual encryption and/or decryption processing of a file. Once the file is decrypted or encrypted the file encryption key is immediately covered for protection.

FCS_CKM_EXP_2.5 The TSF shall destroy non-persistent cryptographic keys after a cryptographic administrator-defined period of time of inactivity.

Application Note: The cryptographic administrator must have the ability to set a threshold of inactivity after which non-persistent keys must be destroyed in accordance with FCS_CKM.4.

.**FCS_CKM_EXP.2.6:** The TSF shall overwrite each intermediate storage area for plaintext key/critical cryptographic security parameter (i.e., any storage, such as memory buffers, that is included in the path of such data). This overwriting shall be performed as follows:

a) For non-volatile memories other than EEPROM and Flash, the overwrite shall be executed three or more times using a different alternating data pattern each time upon the transfer of the key/critical cryptographic security parameter to another location.

b) For volatile memory and non-volatile EEPROM and Flash memories, the overwrite shall be a single direct overwrite consisting of a pseudo random pattern, followed by a read-verify upon the transfer of the key/critical cryptographic security parameter to another location.

Application Note: This is related to the elimination of internal, temporary copies of plaintext keys created during processing, not to the total destruction of a key from the TOE which is discussed under Key Destruction. Although verification of the zeroization of each intermediate location consisting of non-volatile memories, of a plaintext key/critical cryptographic security parameter is desired here (by checking for the final known alternating data pattern), it is not required at this time. However, vendors are highly encouraged to incorporate this verification whenever possible into their implementations.

FCS_CKM_EXP.2.7: The TSF shall prevent archiving of expired (private) signature keys.

Application Note: This requirement is orthogonal to typical system back-up procedures. Therefore, it does not address the problem of archiving an active (private) signature key during a system back-up and saving the key beyond its intended life span.

5.2.3 Explicit: Cryptographic Operations Availability (FCS_COA_EXP)

5.2.3.1 Explicit: Cryptographic Operations Availability (FCS_COA_EXP.1)

FCS_COA_EXP.1 The TSF shall provide the following cryptographic operations to applications:

a) Encryption/Decryption

b) Cryptographic Signature (Digital Signature)

c) Hashing

d) Cryptographic Key Agreement

e) ***[assignment: any other cryptographic operations provided to applications]*****.**

Application Note: Combinations of these operations are also permissible. For instance, an encryption mode such as Galois Counter Mode which provides both encryption and data integrity (which is normally provided via secure hashing), is allowed.

5.2.4 Cryptographic Operation (FCS_COP)

5.2.4.1 Cryptographic Operation (for data encryption/decryption) (FCS_COP.1(1))

FCS_COP.1.1(1) **Refinement:** The TSF shall perform **data encryption/decryption services** in accordance with a **NIST-approved implementation of the Advanced Encryption Standard (AES) cryptographic algorithm used in NIST-approved modes of operation and cryptographic key size of at least 128 bits**, that meets the following: **8**

- **FIPS PUB 197, "Advanced Encryption Standard (AES)"**
- **FIPS PUB 140-2, "Security Requirements for Cryptographic Modules"**
- **NIST Special Publications 800-38A, 800-38B and 800-38C, "Recommendation for Block Cipher Modes of Operation"**

5.2.4.2 Cryptographic Operation (for cryptographic signature) (FCS_COP.1(2))

FCS_COP.1.1(2) **Refinement:** The TSF shall perform **cryptographic signature services** in accordance with a **NIST-approved implementation of** ***[selection:***

(1) Digital Signature Algorithm (DSA) with a key size (modulus) of 2048 bits or greater,

(2) RSA Digital Signature Algorithm (rDSA) with a key size (modulus) of 2048 bits or greater, or

(3) Elliptic Curve Digital Signature Algorithm (ECDSA) with a key size of 256 bits or greater]

Application Note: As the preferred approach for cryptographic signature, elliptic curves will be required after all the necessary standards and other supporting information are fully established.

that meets the following: **9**

a) Case: Digital Signature Algorithm

- **FIPS PUB 186-3, "Digital Signature Standard"**
- **NIST Special Publication 800-57, "Recommendation for Key Management"**

Application Note: Any pseudorandom RNG used in these schemes for generating private values is seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this Protection Profile).

b) Case: RSA Digital Signature Algorithm

- **FIPS PUB 186-3, "Digital Signature Standard"**
- **NIST Special Publication 800-57, "Recommendation for Key Management"**
- **ANSI X9.31-1998 (May 1998), "Digital Signatures Using Reversible Public Key Cryptography For The Financial Services Industry (rDSA)"**

Application Note: Any pseudorandom RNG used in these schemes for generating private values is seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this Protection Profile).

c) Case: Elliptic Curve Digital Signature Algorithm

- **FIPS PUB 186-3, "Digital Signature Standard"**
- **NIST Special Publication 800-57, "Recommendation for Key Management"**
- **Only the "NIST curves" P-256, P-384 and P-521 (as defined in FIPS PUB 186-3, "Digital Signature Standard") may be used.**

Application Note: Any pseudorandom RNG used in these schemes for generating private values is seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this Protection Profile).

5.2.4.3 Cryptographic Operation (for cryptographic hashing) (FCS_COP.1(3))

FCS_COP.1.1(3) **Refinement:** The TSF shall perform **cryptographic hashing services** in accordance with a **NIST-approved implementation of the Secure Hash** algorithm and **message digest size of at least 256 bits** that meets the following**: FIPS PUB 180-2, "Secure Hash Standard".**

Application Note: The message digest size should correspond to double the system encryption key strength.

5.2.4.4 Cryptographic Operation (for cryptographic key agreement) (FCS_COP.1(4))

FCS_COP.1.1(4) **Refinement:** The TSF shall perform **cryptographic key agreement services** in accordance with a **NIST-approved implementation of a *[selection:***

(1) Finite Field-based key agreement algorithm and cryptographic key sizes (modulus) of 2048 bits or greater, or

(2) Elliptic Curve-based key agreement algorithm and cryptographic key size of 256 bits or greater]

Application Note: For elliptic curve-based schemes the key size refers to the log_2 of the order of the base point. As the preferred approach for key exchange, elliptic curves will be required after all the necessary standards and other supporting information are fully established.

that meets the following: **10**

a) Case: Finite field-based key agreement schemes[17]

- **NIST Special Publication 800-57, "Recommendation for Key Management"**
- **NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"**

Application Note: Any pseudorandom RNG used in these schemes for generating private values is seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this Protection Profile).

b) Case: Elliptic curve-based key agreement schemes

- **NIST Special Publication 800-57, "Recommendation for Key Management"**
- **NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"**
- **Only the "NIST curves" P-256, P-384 and P-521 (as defined in FIPS PUB 186-3, "Digital Signature Standard") may be used.**

Application Note: Any pseudorandom RNG used in these schemes for generating private values is seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this Protection Profile).

Application Note: Some authentication mechanism on the keying material is recommended. In addition, repeated generation of the same shared secrets should be avoided.

5.2.4.5 Explicit: Random Number Generation (FCS_COP_EXP.1)

FCS_COP_EXP.1.1 The TSF shall perform **all random number generation (RNG) services** in accordance with a **NIST-approved implementation of an RNG** that meets the following:

[17] For example, "classic" Diffie-Hellman-based schemes.

a) NIST Special Publication 800-90, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators".

b) The random number generator shall be seeded by *[selection:*

(1) one or more independent hardware-based entropy sources, and/or

(2) one or more independent software-based entropy sources, and/or

(3) a combination of hardware-based and software-based entropy sources.]

c) Additionally, the random number generator shall meet the following:

- **Statistical RNG tests listed in Appendix C**
- **RNG/PRNG design and test documentation consistent with that required in this Protection Profile for other subsystems: Development Documentation (ADV).**

Application Note: When the ANSI X9.82 "Random Number Generation" standard is finalized, it will be incorporated here.

Application Note: Successful completion and documentation of these tests during the TOE development helps to demonstrate the RNG design is rigorous. There exists a NIST toolbox for running these tests. Requirements for acceptable thresholds and sample sizes for use in applying NIST Special Publication 800-22 in the context of this protection profile can be found in Appendix D. Note that the Appendix D tests are just demonstrated once during development and are NOT to be done "at power up, conditionally, on demand, or periodically"

FCS_COP_EXP.1.2 The TSF shall defend against tampering of the random number generation (RNG)/ pseudorandom number generation (PRNG) sources.

Application Note: The RNG/PRNG should be resistant to manipulation or analysis of its sources, or any attempts to predictably influence its states. Three examples of very different approaches the TSF might pursue to address this include: a) identifying the fact that physical security must be applied to the product, b) applying checksums over the sources, or c) designing and implementing the TSF RNG with a concept similar to a keyed hash (e.g., where periodically, the initial state of the hash is changed unpredictably and each change is protected as when provided on a tamper-protected token, or in a secure area of memory.

5.3 User Data Protection (FDP)

5.3.1 Access Control Policy (FDP_ACC)

5.3.1.1 Complete Access Control (FDP_ACC.2)

FDP_ACC.2.1 The TSF shall enforce the **Discretionary Access Control policy** on ***all subjects and all named objects*** and all operations among **them**.[11]

Application Note: The DAC policy does not cover local public objects.

FDP_ACC.2.2 **Refinement:** The TSF shall ensure that all operations between any subject and any **named** object are covered by **the Discretionary Access Control policy**.[12]

5.3.2 Access Control Functions (FDP_ACF)

5.3.2.1 Security Attribute Based Access Control (FDP_ACF.1)

FDP_ACF.1.1 **Refinement:** The TSF shall enforce the **Discretionary Access Control policy** to **named** objects based on the following **types of subject and object security attributes:**

a) the authorized user identity and group membership(s) associated with a subject and

b) the [authorized user (or group) identity, access operations] pairs associated with a named object.

Application Note: This requirement is worded to include only implementations where access control attributes are associated with objects rather than subjects. This implementation becomes critical when satisfying FMT_REV.1.1(1).

FDP_ACF.1.2 **Refinement:** The TSF shall enforce the following rules to determine if an operation among subjects and **named** objects is allowed:[13]

- **The Discretionary Access Control policy mechanism shall, either by explicit authorized user action or by default, provide that named objects are protected from unauthorized access according to the following ordered rules:**

1) If the requested mode of access is denied to that authorized user, deny access.

2) If the requested mode of access is permitted to that authorized user, permit access.

3) If the requested mode of access is denied to every group of which the authorized user is a member, deny access

4) If the requested mode of access is permitted to any group of which the authorized user is a member, grant access

5) Else deny access.

Application Note: This element specifies minimum granularity of access control functionality. It is not meant to preclude more fine grained access control mechanisms. However any more fine grained mechanisms must be capable of meeting the above rules. For example, discretionary access rules on a file may be defined to take precedence over discretionary access rules on the directories containing that file.

FDP_ACF.1.3 **Refinement:** The TSF shall explicitly authorize access of subjects to **named** objects based on the following additional rules:

a) Authorized administrators must follow the above-stated Discretionary Access Control policy, except after taking the following specific actions: *[assignment:*

***list of specific actions]*.**

b) The enforcement mechanism (i.e., access control lists) shall allow authorized users to specify and control sharing of named objects by individual user identities and group identities and shall provide controls to limit propagation of access rights.

c) [assignment: other rules, based on security attributes, that explicitly authorize access of subjects to **named** *objects]*.

Application Note: This element allows specifications of additional rules for authorized administrators to bypass the Discretionary Access Control policy for system management or maintenance (e.g., system backup).

FDP_ACF.1.4 **Refinement:** The TSF shall explicitly deny access of subjects to **named** objects based on the **following rules**:

a) If the requested mode of access is denied to that authorized user, deny access.

b) If the requested mode of access is denied to every group of which the authorized user is a member, deny access

5.3.3 Residual Information Protection (FDP_RIP)

5.3.3.1 Full Residual Information Protection (FDP_RIP.2)

FDP_RIP.2.1 **Refinement:** The TSF shall ensure that any previous information content of a resource is made unavailable upon the *[selection: allocation of the resource to, deallocation of the resource from]* all objects **other than those associated with cryptographic keys and critical cryptographic security parameters as described in FCS_CKM.4.1 and FCS_CKM_EXP.2.5**.

Application Note: This requirement applies to all resources except for cryptographic keys and critical cryptographic security parameters governed by or used by the TSF; it includes resources used to store data and attributes. It also includes the encrypted representation of information. Residual information protection for cryptographic data is covered in class FCS.

Application Note: Clearing the content of resources on deallocation is sufficient to satisfy this requirement, provided that unallocated resources will not accumulate new information until they are allocated again.

5.4 Identification and Authentication (FIA)

5.4.1 Authentication Failures (FIA_AFL)

5.4.1.1 Authentication Failure Handling (FIA_AFL.1)

FIA_AFL.1.1 The TSF shall detect when **an authorized administrator configurable positive integer of consecutive** unsuccessful authentication attempts occur related to **any authorized user authentication process**.

FIA_AFL.1.2 **Refinement:** When the defined number of **consecutive** unsuccessful authentication attempts has been met or surpassed, the TSF shall:

a) For all administrator accounts, disable the account for an authorized administrator configurable time period;

Application Note: The administrator account is still subject to the delay captured in item 'c' in FIA_SOS.1.

b) For all other accounts, disable the user logon account until it is re-enabled by the authorized administrator.

Application Note: The ability to disable user accounts is necessary to counter brute force discovery of the authentication data.

c) For all disabled accounts, respond with an "account disabled" message without attempting any type of authentication.

Application Note: The actual content of the "account disabled" message can vary. It need not be explicitly "account disabled". The important aspect of the requirement is that no authentication attempt occurs for disabled accounts.

Application Note: "Consecutive unsuccessful authentication attempts" is the total number of unsuccessful attempts that occur, in order, prior to a successful authentication attempt. For distributed systems, the TOE must reconcile unsuccessful attempts across nodes in accordance with FPT_TRC_EXP.1.

5.4.2 User Attribute Definition (FIA_ATD)

5.4.2.1 User Attribute Definition (FIA_ATD.1)

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users:

a) unique identifier;

b) group memberships;

c) authentication data;

d) security-relevant roles (see FMT_SMR.2);

***e) [Assignment: Any security attributes related to cryptographic function (e.g., certificate used to represent the user)]*; and**

f) [Assignment: Any other security-relevant authorizations or attributes (e.g., privilege)].

Application Note: Group membership may be expressed in a number of ways: a list per user specifying to which groups the user belongs, a list per group which includes which users are members, or implicit association between certain user identities and certain groups.

Application Note: A TOE may have two forms of user and group identities which have a unique mapping between the representations.

Application Note: It is possible that the notion of privilege is tied to the security-relevant roles (item d).

5.4.3 Specification of Secrets (FIA_SOS)

5.4.3.1 Verification of Secrets (FIA_SOS.1)

FIA_SOS.1.1 The TSF shall provide a mechanism to verify that secrets meet **the following:**

a) For each attempt to use the authentication mechanism, the probability that a random attempt will succeed is less than one in 5×10^{15};

Application Note: This can be achieved with a password of eight characters, assuming an alphabet of 92 characters.

b) The authentication mechanism must provide the capability for an administrator to specify the conditions that need to be met before an individual user can reuse a secret;

c) For all administrator accounts, the authentication mechanism must provide a delay such that there can be no more than ten attempts per minute; and

Application Note: The administrator account is still subject to the configurable lockout delay in item 'a' in FIA_AFL.1. User accounts are subject to the lockout requirement in item 'b' in FIA_AFL.1.

d) Any feedback given during an attempt to use the authentication mechanism will not reduce the probability below the above metrics.

Application Note: The ST specifies the method of authentication. Where authentication is provided by a password mechanism, the ST shows that the restrictions upon passwords (length, alphabet, conditions for reuse (e.g., time period, number of intermediate secrets), and other characteristics) result in a password space conforming to items (a) and (b) above, as well as characterize the delay to show conformance to item (c) above. Where authentication is provided by a mechanism other than passwords, the ST shows the authentication method has a low probability equivalent to item (a) above that authentication data can be forged or guessed.

5.4.4 User Authentication (FIA_UAU)

5.4.4.1 Timing of Authentication (FIA_UAU.1)

FIA_UAU.1.1 **Refinement:** The TSF shall allow **read access to *public objects*** on behalf of the user to be performed before the user is authenticated.

FIA_UAU.1.2 **Refinement:** The TSF shall require each user to be successfully authenticated **(i.e., an exact match between the internal representation of the user's entered data and the stored TSF authentication data)** before allowing any other TSF-mediated actions on behalf of that user.

Application Note: The entire entered user's authentication data must exactly match the entire

stored data. No other parameters such as length of password should be used to short-circuit the authentication verification.

5.4.4.2 Re-authenticating (FIA_UAU.6)

FIA_UAU.6.1 **Refinement:** The TSF shall re-authenticate the user **when changing authentication data**.[14]

Application Note: If the TOE is requiring the user to change authentication data upon having just authenticated (e.g., initial logon, session unlock), the user is considered to be re-authenticated.

5.4.4.3 Protected Authentication Feedback (FIA_UAU.7)

FIA_UAU.7.1 The TSF shall provide only **obscured feedback** to the user while the authentication is in progress.

Application Note: "Obscured feedback" implies the TSF does not produce a visible display of any authentication data entered by a user (such as the echoing of a password), although an obscured indication of progress may be provided (such as an asterisk for each character). It also implies that the TSF does not return any information during the authentication process to the user, which may provide any indication of the authentication data.

5.4.5 User Identification (FIA_UID)

5.4.5.1 Timing of Identification (FIA_UID.1)

FIA_UID.1.1 The TSF shall allow **read access to *public objects*** on behalf of the user to be performed before the user is identified.

FIA_UID.1.2 The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

5.4.6 User-Subject Binding (FIA_USB)

5.4.6.1 User-Subject Binding (FIA_USB.1)

FIA_USB.1.1 The TSF shall associate the following user security attributes with subjects acting on behalf of that user:

a) The unique user identity that is associated with auditable events;

b) The user identity or identities that are used to enforce the Discretionary Access Control Policy;

Application Note: The DAC and audit policies require that each subject acting on behalf of a user has a user identity associated with the subject. While this identity is typically the one used at the time of identification to the system, the DAC policy enforced by the TSF may include provisions for making access decisions based upon a different user identity, such as the "set user ID (su)" command in UNIX.

c) The group identity or identities that are used to enforce the Discretionary Access Control Policy; and

d) The user's authorized roles;

e) [Assignment: other list of user security attributes related to cryptographic function (e.g., certificate used to represent the user, key used to encrypt data on behalf of the user)].

Application Note: The attributes listed in FIA_USB.1 should be comparable to those listed in FIA_ATD.1.

5.5 Security Management (FMT)

5.5.1 Management of Functions in TSF (FMT_MOF)

5.5.1.1 Management of Security Functions Behavior (for specification of auditable events) (FMT_MOF.1(1))

FMT_MOF.1.1(1) **Refinement:** The TSF shall restrict the ability to <u>disable and enable</u> the **audit** functions **and to specify which events are to be audited (see FAU_SEL.1.1)** to **the authorized administrators.**

Application Note: To "specify" means the ability to select what events will be audited.

5.5.1.2 Management of Security Functions Behavior (for authentication data) (FMT_MOF.1(2))

FMT_MOF.1.1(2) **Refinement:** The TSF shall restrict the ability to **manage the values of security attributes associated with user authentication data** to **authorized administrators.**[15]

Application Note: The word "manage" includes but is not limited to create, initialize, change default, modify, delete, clear, append, and query. Security attributes associated with user authentication data include password length, expiration, history, etc.

5.5.2 Management of Security Attributes (FMT_MSA)

5.5.2.1 Management of Security Attributes (FMT_MSA.1(1))

FMT_MSA.1.1(1) **Refinement:** The TSF shall enforce the **Discretionary Access Control policy** to restrict the ability to **<u>change</u>** the **value of object** security attributes **to authorized administrators and owners of the object.**[16]

5.5.2.2 Management of Security Attributes (for Object Ownership) (FMT_MSA.1(2))

FMT_MSA.1.1(2) **Refinement:** The TSF shall enforce the **Discretionary Access Control policy** to restrict the ability to **<u>change</u> object ownership to**

authorized administrators.[17]

Application Note: This requirement prevents a user from changing object ownership to another user

5.5.2.3 Secure Security Attributes (FMT_MSA.2)

FMT_MSA.2.1 **Refinement:** The TSF shall ensure that only **valid** values are accepted for security attributes.[18]

Application Note: Valid implies that the values fall within an appropriate range for that attribute (e.g., the password length attribute must be a non-negative integer).

5.5.2.4 Static Attributes Initialization (FMT_MSA.3)

FMT_MSA.3.1 The TSF shall enforce the **Discretionary Access Control policy** to provide restrictive default values for security attributes that are used to enforce the SFP.

Application Note: The TOE must provide protection by default for all objects at creation time. This may allow authorized users to explicitly specify the desired access controls upon the object at its creation, provided that there is no window of vulnerability through which unauthorized access may be gained to newly-created objects.

FMT_MSA.3.2 The TSF shall allow the **authorized administrator** to specify alternative initial values to override the default values when an object or information is created.

Application Note: This requirement applies as a system-wide default, However, users may be allowed to define default values for objects they create (e.g., per user or per object type).

5.5.3 Management of TSF Data (FMT_MTD)

5.5.3.1 Management of TSF Data (for general TSF data) (FMT_MTD.1(1))

FMT_MTD.1.1(1) The TSF shall restrict the ability to **manage** the **security-relevant TSF data except for audit records, user security attributes, authentication data, and critical cryptographic security parameters** to **authorized administrators**.

Application Note: The word "manage" includes but is not limited to create, initialize, change default, modify, delete, clear, append, and query. Security attributes associated with user authentication data include password length, password expiration, password history, etc. The restrictions for audit records, user security attributes, authentication data, and critical cryptographic security parameters are specified below.

5.5.3.2 Management of TSF Data (for audit data) (FMT_MTD.1(2))

FMT_MTD.1.1(2) The TSF shall restrict the ability to query, delete, and clear the **audit records** to **authorized administrators**.

Application Note: This requirement applies to actions taken on the entire audit file/log, not actions on individual audit records.

5.5.3.3 Management of TSF Data (for initialization of user security attributes) (FMT_MTD.1(3))

FMT_MTD.1.1(3) The TSF shall restrict the ability to **initialize user security attributes** to **authorized administrators**.

5.5.3.4 Management of TSF Data (for modification of user security attributes, other than authentication data) (FMT_MTD.1(4))

FMT_MTD.1.1(4) The TSF shall restrict the ability to modify **user security attributes, other than authentication data,** to **authorized administrators.**

5.5.3.5 Management of TSF Data (for modification of authentication data) (FMT_MTD.1(5))

FMT_MTD.1.1(5) The TSF shall restrict the ability to modify **authentication data** to **authorized administrators and users authorized to modify their own authentication data.**

5.5.3.6 Management of TSF Data (for reading of authentication data) (FMT_MTD.1(6))

FMT_MTD.1.1(6) **Refinement:** The TSF shall **prevent reading of authentication data**.[19]

5.5.3.7 Management of TSF Data (for critical cryptographic security parameters) (FMT_MTD.1(7))

FMT_MTD.1.1(7) The TSF shall restrict the ability to **manage** the **critical cryptographic security parameters and data related to cryptographic configuration** to **cryptographic administrators**.

Application Note: The word "manage" includes but is not limited to create, initialize, change default, modify, delete, clear, append, and query. Critical cryptographic security parameters are defined in the glossary where examples are also provided. Examples of data related to cryptographic configuration include, but are not limited to: setting of the cryptographic algorithm, setting the cryptographic mode of operation, setting the key length, setting a hash digest size, etc."

5.5.4 Revocation (FMT_REV)

5.5.4.1 Revocation (to authorized administrators) (FMT_REV.1(1))

FMT_REV.1.1(1) The TSF shall restrict the ability to revoke security attributes associated with the users within the TSC to **authorized administrators**.

Application Note: The phrase "revoke security attributes" means to change attributes so that

access is revoked.

FMT_REV.1.2(1) **Refinement:** The TSF shall enforce the **immediate revocation of security-relevant authorizations.**[20]

Application Note: Security-relevant authorizations include the ability of authorized users to log in or perform privileged operations. An example of revoking a security-relevant authorization is the deletion of a user account upon which system access is immediately terminated.

5.5.4.2 Revocation (to owners and authorized administrators) (FMT_REV.1(2))

FMT_REV.1.1 (2) **Refinement:** The TSF shall restrict the ability to revoke security attributes **of named** objects within the TSC to **owners of the named object and authorized administrators**.[21]

Application Note: The term "revoke security attributes" means "change attributes so that access is revoked".

FMT_REV.1.2 (2) **Refinement:** The TSF shall enforce the **revocation of access rights associated with named objects when an access check is made.**[22]

Application Note: The state where access checks are made determines when the access control policy enforces revocation. The access control policy may include immediate or delayed revocation. The access rights are considered to have been revoked when all subsequent access control decisions made by the TSF use the new access control information. In cases where a previous access control decision was made to permit an operation, it is not required that every subsequent operation make an explicit access control decision.

5.5.5 Security Attribute Expiration (FMT_SAE)

5.5.5.1 Time-Limited Authorization (FMT_SAE.1)

FMT_SAE.1.1 The TSF shall restrict the capability to specify an expiration time for **authorized user authentication data** to **the authorized administrator**.

FMT_SAE.1.2 **Refinement:** The TSF shall be able to **lock out the associated authorized user account** after the expiration time has passed. [23]

5.5.6 Security Management Roles (FMT_SMR)

5.5.6.1 Security Roles (FMT_SMR.2)

FMT_SMR.2.1 The TSF shall maintain the roles:

a) authorized administrator;

Application Note: Any user that is authorized to modify the TOE such that the DAC policy is bypassed is by definition, an authorized administrator. The TOE may provide multiple administrator roles (audit administrator, security administrator, etc).

b) cryptographic administrator

Application Note: Any user authorized to perform functions that affect the operation of the cryptographic module(s) such as cryptographic initialization, setting of cryptographic algorithm modes, and selection of the algorithms is by definition, a cryptographic administrator.

c) [assignment: any other mutually exclusive roles that are derived from a proper subset of the above roles].

Application Note: If an additional role is defined, it must be derived from a proper subset of one of the above roles because they encompass all security relevant administrative functionality. The associated requirements must be appropriately refined such that the new role is mutually exclusive from all other roles. For example, creating an audit administrator role from a subset of the authorized administrator role would require refining all requirements related to audit (e.g., FMT_MTD.1.1(2)) to state "audit administrator" vice "authorized administrator".

FMT_SMR.2.2 **Refinement:** The TSF shall be able to associate **authorized** users with roles.

FMT_SMR.2.3 **Refinement:** The TSF shall ensure that **roles are distinct and that no overlap of allowed operations exists between roles**.[24]

Application Note: If new roles are defined under FMT_SMR.2.1 item 'c' above, the new role is no longer a subset of the PP defined roles in items 'a' and 'b'. Furthermore, the functions of the new role are no longer part of the PP defined role.

5.5.6.2 Assuming Roles (FMT_SMR.3)

FMT_SMR.3.1 **Refinement**: The TSF shall require an explicit request to assume **any role**.

5.5.7 Specification of Management Functions (FMT_SMF)

5.5.7.1 Specification of Management Functions (FMT_SMF.1)

FMT_SMF.1.1 The TSF shall be capable of performing the following security management functions: **all security management functions identified in other sections of this PP**.

Application Note: The security management functions for FMT_SMF.1 are distributed throughout the PP and are included as part of the requirements in FMT_MOF, FMT_MSA, FMT_MTD, FMT_REV, FMT_SAE, FPT_TST, FRU_RSA and any cryptographic management functions specified in the reference standards. Compliance to these requirements satisfies compliance with FPT_SMF.1.

5.6 Protection of the TOE Security Functions (FPT)

5.6.1 Internal TOE TSF Data Transfer (FPT_ITT)

5.6.1.1 Basic Internal TSF Data Transfer Protection (FPT_ITT.1)

FPT_ITT.1.1 **Refinement:** The TSF shall protect TSF data from disclosure when it is transmitted between separate parts of the TOE **through the use of the TSF-provided cryptographic services**.

Application Note: Refer to FCS_COP.1.1(1) for TSF-provided cryptographic services

5.6.1.2 TSF Data Integrity Monitoring (FPT_ITT.3)

FPT_ITT.3.1 **Refinement:** The TSF shall be able to detect modification and **insertion** of TSF data transmitted between separate parts of the TOE **through the use of the TSF-provided cryptographic services**.

Application Note: Use of a keyed hash function (e.g., HMAC) that is: (1.) calculated over the TSF data to be transmitted, (2.) appended to the transmitted TSF data, and (3.) checked by the receiving part of the TOE is an example of a cryptographic means that detects modification and substitution of such data. Another example is the use of a cryptographic signature over the transmitted TSF data

Application Note: Refer to FCS_COP.1.1(2) and FCS_COP.1.1(3) for TSF-provided cryptographic services.

FPT_ITT.3.2 Upon detection of a data integrity error, the TSF shall take the following actions:

a) reject data

b) audit event

c) [assignment: specify the action to be taken].

Application Note: Additional actions ST author might consider are: retransmission of data and, an alarm after reaching a retransmission threshold.

5.6.2 Trusted Recovery (FPT_RCV)

5.6.2.1 Manual Recovery (FPT_RCV.1)

FPT_RCV.1.1 **Refinement:** After **a failure or service discontinuity that may lead to a violation of the TSP**, the TSF shall enter a maintenance mode where the ability to return **the TOE** to a secure state is provided. **As part of the secure state, the cryptographic module shall be in a known and secure state such that all storage is empty of plaintext cryptographic keys and sensitive data and inaccessible to processes, and all security policies are enforced**.

Application Note: In maintenance mode normal operation might be impossible or severely

restricted, as otherwise insecure situations might occur. Typically, only authorized users should be allowed access to this mode.

5.6.3 Replay Detection

5.6.3.1 Replay Detection (FPT_RPL.1)

FPT_RPL.1.1 **Refinement:** The TSF shall **be able to** detect replay **of TSF data transmitted between separate parts of the TOE through the use of the TSF-provided cryptographic services**.**25**

Application Note: Use of a keyed hash function (e.g., HMAC) that is: (1.) calculated over the TSF data to be transmitted, (2.) appended to the transmitted TSF data, and (3.) checked by the receiving part of the TOE is an example of a cryptographic means that detects modification and substitution of such data. Another example is the use of a cryptographic signature over the transmitted TSF data.

Application Note: Refer to FCS_COP.1.1(2) and FCS_COP.1.1(3) for TSF-provided cryptographic services .

FPT_RPL.1.2 **Refinement:** Upon detection of **TSF data** replay, the TSF shall take the following actions:**26**

a) reject data

b) audit event

c) [assignment: specify the action to be taken].

Application Note: Additional actions ST author might consider are: retransmission of data and, an alarm after reaching a retransmission threshold.

5.6.4 Reference Mediation (FPT_RVM)

5.6.4.1 Non-Bypassability of the TSF (FPT_RVM.1)

FPT_RVM.1.1 The TSF shall ensure that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed.

5.6.5 Domain Separation (FPT_SEP)

5.6.5.1 SFP Domain Separation (FPT_SEP.2)

FPT_SEP.2.1 **Refinement:** The **non-cryptography** portion of the TSF shall maintain a security domain for its own execution that protects it from interference and tampering by untrusted subjects.

FPT_SEP.2.2 The TSF shall enforce separation between the security domains of subjects in the TSC.

FPT_SEP.2.3 **Refinement:** The TSF shall maintain **separation of** the part of the TSF related to **cryptography**[18] that protects it from interference and tampering by the remainder of the TSF and by subjects untrusted with respect to **cryptography**.**27**

Application Note: Although not required at this time, establishing a separate address space for the cryptography for its own execution and that protects it from accidental interference and tampering by malicious untrusted subjects is the preferred approach for meeting this requirement in medium robustness products, and will be required in updated versions of the OS PP in the near future. For now, as an interim solution, other combinations of techniques that jointly support the overall protection and logical separation of the cryptography may be acceptable pending NSA review.

Application Note: Ideally, use of off board hardware or a third processor hardware state is the most preferred implementation supporting separation, because it would protect the cryptography from all other parts of the TSF, including malicious parts of the kernel. Migration to this most preferred implementation is anticipated eventually.

5.6.6 Time Stamps (FPT_STM)

5.6.6.1 Reliable Time Stamps (FPT_STM.1)

FPT_STM.1.1 The TSF shall be able to provide reliable time stamps.

Application Note: A time stamp includes the correct date and time such that the order of events can be determined.

5.6.7 Internal TOE TSF Data Replication Consistency (FPT_TRC)

5.6.7.1 Explicit: Internal TSF Data Consistency (FPT_TRC_EXP.1)

FPT_TRC_EXP.1.1 The TSF shall ensure that TSF data is consistent between parts of the TOE by providing a mechanism to bring inconsistent TSF data into a consistent state in a timely manner.

Application Note: In general, it is impossible to achieve complete, constant consistency of TSF data that is distributed to remote portions of a TOE because distributed portions of the TSF may be active at different times or disconnected from one another. This requirement attempts to address this situation in a practical manner by acknowledging that there will be TSF data inconsistencies but that they will be corrected without undue delay. For example, a TSF could provide timely consistency through periodic broadcast of TSF data to all TSF nodes maintaining replicated TSF data. Another example approach is for the TSF to provide a mechanism to explicitly probe remote TSF nodes for inconsistencies and respond with action to correct the identified inconsistencies.

[18] At a minimum this separation must be maintained for the part of the TSF implementing the cryptoalgorithm and the management of persistent keys.

5.6.8 TSF Self Testing (FPT_TST)

5.6.8.1 Explicit: TSF Testing (FPT_TST_EXP.1)

FPT_TST_EXP.1.1 The TSF shall run a suite of self tests during the initial start-up and also either periodically during normal operation, or at the request of an authorized administrator to demonstrate the correct operation of the TSF.

FPT_TST_EXP.1.2 The TSF shall provide authorized administrators with the capability to verify the integrity of stored TSF executable code through the use of the TSF-provided cryptographic services.

Application Note: Refer to FCS_COP.1.1(2) and FCS_COP.1.1(3) for TSF-provided cryptographic services .

5.6.8.2 TSF Testing (for cryptography) (FPT_TST.1(1))

FPT_TST.1.1(1) **Refinement:** The TSF shall run a suite of self tests **in accordance with FIPS PUB 140-2, Level 4 and Appendix C of this profile** during initial start-up **(on power on)**, at the request of the **cryptographic administrator (on demand)**, **under various conditions defined in section 4.9.1 of FIPS 140-2,** and periodically **(at least once a day)** to demonstrate the correct operation of the **following cryptographic functions:**[28]

a) key error detection;

b) cryptographic algorithms;

c) RNG/PRNG;

Application Note: These tests apply regardless of whether the cryptographic functionality is implemented in hardware, software, or firmware.

FPT_TST.1.2(1) **Refinement:** The TSF shall provide authorized **cryptographic administrators** with the capability to verify the integrity of **TSF data related to the cryptography by using TSF-provided cryptographic functions**.[29]

Application Note: Refer to FCS_COP.1.1(2) and FCS_COP.1.1(3) for TSF-provided cryptographic services

FPT_TST.1.3(1) **Refinement:** The TSF shall provide authorized **cryptographic administrators** with the capability to verify the integrity of stored TSF executable code **related to the cryptography by using TSF-provided cryptographic functions**.[30]

Application Note: Refer to FCS_COP.1.1(2) and FCS_COP.1.1(3) for TSF-provided cryptographic services

5.6.8.3 TSF Testing (for key generation components) (FPT_TST.1(2))

FPT_TST.1.1(2) **Refinement**: The TSF shall **perform** self tests **immediately after generation of a key** to demonstrate the correct operation **of each key generation component**. **If any of these tests fails, that generated key shall not be used, the cryptographic module shall react as required by FIPS PUB 140-2 for failing a self-test, and this event will be audited**.**31**

Application Note: Key generation components are those critical elements that compose the entire key generation process (e.g., any algorithms, any RNG/PRNGs, any key generation seeding processes, etc.).

Application Note: These self-tests on the key generation components can be executed here as a subset of the full suite of self-tests run on the cryptography in FPT_TST.1(1) as long as all elements of the key generation process are tested.

FPT_TST.1.2(2) **Refinement:** The TSF shall provide authorized **cryptographic administrators** with the capability to verify the integrity of TSF data **related to the key generation by using TSF-provided cryptographic functions**.**32**

Application Note: Refer to FCS_COP.1.1(2) and FCS_COP.1.1(3) for TSF-provided cryptographic services

FPT_TST.1.3(2) **Refinement:** The TSF shall provide authorized **cryptographic administrators** with the capability to verify the integrity of stored TSF executable code **related to the key generation by using TSF-provided cryptographic functions**.**33**

Application Note: Refer to FCS_COP.1.1(2) and FCS_COP.1.1(3) for TSF-provided cryptographic services

5.7 Resource Utilization (FRU)

5.7.1 Resource Allocation (FRU_RSA)

5.7.1.1 Maximum Quotas (for shared persistent storage) (FRU_RSA.1(1))

FRU_RSA.1.1(1) The TSF shall enforce maximum quotas of the following resources: **portion of shared persistent storage** that individual **authorized** users can use simultaneously.

Application Note: For persistent storage, simultaneously means that the shared media contains data belonging to more than one user.

5.7.1.2 Maximum Quotas (for system memory) (FRU_RSA.1(2))

FRU_RSA.1.1(2) The TSF shall enforce maximum quotas of the following resources: **portion of system memory** that individual **authorized** users can use

simultaneously.

5.7.1.3 Maximum Quotas (for processing time) (FRU_RSA.1(3))

FRU_RSA.1.1(3) The TSF shall enforce maximum quotas of the following resources: **portion of processing time** that subjects can use over a specified period of time.

Application Note: The algorithm to determine percentages of time can be based on many factors (e.g., number of users, relative priority of users, availability of resources to users).

5.8 TOE Access (FTA)

5.8.1 Limitation on scope of selectable attributes (FTA_LSA)

5.8.1.1 Limitation on scope of selectable attributes (FTA_LSA.1)

FTA_LSA.1.1 **Refinement**: The TSF shall restrict the scope of **roles and user privileges** based on location, time, and day.**34**

Application Note: The intent of this requirement is to allow or disallow the assumption of roles or the effectiveness of user privileges based on the location where the session was established or the date/time of session establishment.

Application Note: "Location" refers to what ever means the TOE uses to identify a point of entry for interactive user session establishment. The adequacy of this means is determined by other requirements (e.g., FPT_SEP, AVA_VLA).

5.8.2 Limitation on multiple concurrent sessions (FTA_MCS)

5.8.2.1 Basic limitation on multiple concurrent sessions (FTA_MCS.1)

FTA_MCS.1.1 **Refinement:** The TSF shall **enforce a** maximum number of concurrent **interactive** sessions per user.**35**

FTA_MCS.1.2 **Refinement:** The TSF shall allow **an authorized administrator to set the maximum number of concurrent interactive** sessions per user.**36**

Application Note: In distributed TOE implementations where synchronization of TSF data is a concern, the internal TSF data consistency requirement FPT_TRC_EXP.1.1 applies and any violations of the above requirement must be remedied at every synchronization.

5.8.3 Session Locking (FTA_ SSL)

5.8.3.1 TSF-Initiated Session Locking (FTA_SSL.1)

FTA_SSL.1.1 The TSF shall lock an interactive session after **an authorized administrator specified time interval of user inactivity** by:

a) clearing or overwriting display devices, making the current contents unreadable.

b) disabling any activity of the user's data access/display devices other than unlocking the session.

FTA_SSL.1.2 **Refinement:** The TSF shall require the **user to re-authenticate** to unlock the session**.37**

5.8.3.2 User-Initiated Locking (FTA_SSL.2)

FTA_SSL.2.1 The TSF shall allow user-initiated locking of the user's own interactive session by:

a) clearing or overwriting display devices, making the current contents unreadable.

b) disabling any activity of the user's data access/display devices other than unlocking the session.

FTA_SSL.2.2 **Refinement:** The TSF shall require the **user to re-authenticate** to unlock the session.**38**

5.8.4 TOE Access Banners (FTA_TAB)

5.8.4.1 Default TOE Access Banners (FTA_TAB.1)

FTA_TAB.1.1 **Refinement:** Before establishing a user session, the TSF shall display an **authorized-administrator specified** advisory **notice and consent** warning message regarding unauthorized use of the TOE.

5.8.5 TOE Access History (FTA_TAH)

5.8.5.1 TOE Access History (FTA_TAH.1)

FTA_TAH.1.1 **Refinement:** Upon successful **interactive** session establishment, the TSF shall display **to the authorized user** the date and time of **that authorized user's** last successful **interactive** session establishment.

FTA_TAH.1.2 **Refinement:** Upon successful **interactive** session establishment, the TSF shall display **to the authorized user** the date and time of the last unsuccessful attempt and the number of unsuccessful attempts at **interactive** session establishment **for that user identifier** since the last successful **interactive** session establishment.

Application Note: In both of the above elements, for distributed systems, date and time needs to be accurate to the degree required by FPT_TRC_EXP.1.

FTA_TAH.1.3 **Refinement:** The TSF shall not erase the access history information from the **authorized** user interface without giving the **authorized** user the opportunity to review the information.

5.8.6 TOE Session Establishment (FTA_TSE)

5.8.6.1 TOE Session Establishment (FTA_TSE.1)

FTA_TSE.1.1 The TSF shall be able to deny session establishment based on **location, time, and day**.

5.9 Trusted Path/Channels (FTP)

5.9.1 Trusted Path (FTP_TRP)

5.9.1.1 Explicit: Trusted Path (FTP_TRP_EXP.1)

FTP_TRP_EXP.1.1 The TSF shall provide a communication path between itself and remote and local users that is logically distinct from other communication paths and provides assured identification of the TSF to the requesting user and protection of the communicated data from disclosure or undetected modification.

Application Note: This "distinct" path is merely invoked for the duration of its being needed (e.g., for reauthenticating the user); it need not be invoked for the duration of the user's session.

FTP_TRP_EXP.1.2 The TSF shall permit local users and remote users to initiate communication via the trusted path.

FTP_TRP_EXP.1.3 The TSF shall require the use of the trusted path for user authentication and user identification during TOE session establishment, for operations to modify authentication data, for protection of authentication data when a locked session is being unlocked and all other operations requiring a human user to enter authentication data.

End Notes

This section records the functional requirements where deletions of Common Criteria text were performed.

---

1 A deletion of CC text was performed in FAU_ARP.1.1. Rationale: The word "take" was deleted for clarity and better flow of the requirement. Additionally the words, “upon detection of a potential security violation” were moved to the beginning of the requirement to make requirement clearer.

FAU_ARP.1.1 **Refinement:** Upon detection of a potential security violation, the TSF shall ~~take~~ **generate a warning to the authorized administrator** ~~upon detection of a potential security violation~~ **that requires explicit acknowledgement by the administrator.**

2 A deletion of CC text was performed in FAU_SAR.1.2. Rationale: The word "user" was deleted to replace it with "authorized administrator". By default, authorized administrators are the only users with read access to audit records unless granted explicit read-access (FAU_SAR.2).

FAU_SAR.1.2 **Refinement:** The TSF shall provide the audit records in a manner suitable for the ~~user~~ **authorized administrator** to interpret the information **using a tool to access the audit records**.

3 A deletion of CC text was performed in FAU_STG.1.2. Rationale: The word “unauthorized” was from “unauthorized administrator”, since no one can be authorized to modify audit records.

FAU_STG.1.2 **Refinement:** The TSF shall be able to prevent ~~unauthorized~~ modifications to the audit records.

4 A deletion of CC text was performed in FAU_STG.4.1. Rationale: The words " user with special rights”. “User with special rights” was replaced with “authorized administrator” inside the selection. The phrase “if the audit trail is full” was moved to the beginning of the element and changed to say “When the audit trail becomes full”. All these changes were made to make requirement more clear and for better flow.

FAU_STG.4.1 - **Refinement**: When the audit trail becomes full, the TSF shall **provide the authorized administrator the capability to** prevent auditable events , except those taken by the authorized ~~user with special rights~~ **administrator (in the context of performing TOE maintenance)** and **generate an alarm to the authorized administrator**~~, if the audit trail is full~~.

5 A deletion of CC text was performed in FCS_CKM.1.1(1). Rationale: The words “*[assignment: cryptographic key generation algorithm]*” and "and specified cryptographic key sizes [*assignment: cryptographic key sizes*]" were deleted for clarity and better flow of the requirement. The symmetric key generation uses a random number generator that can be implemented in a number of way and using different schemes. By deleting the CC words, the element better states the intended requirement.

FCS_CKM.1.1(1) - **Refinement:** The TSF shall generate **symmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm ~~*[assignment: cryptographic key generation algorithm]*~~ **as follows**: ***[selection:***

(1) a hardware random number generator (RNG) as specified in FCS_COP_EXP.1, and/or

(2) a software random number generator (RNG) as specified in FCS_COP_EXP.1, and/or

(3) a key establishment scheme as specified in FCS_COP.1(4) based upon public key cryptography using a RNG as specified in FCS_COP_EXP.1, and/or a hardware RNG as specified in FCS_COP_EXP.1]

~~and specified cryptographic key sizes [*assignment: cryptographic key sizes*]~~ that meets the following …

6 A deletion of CC text was performed in FCS_CKM.1.1(2). Rationale: The words "specified cryptographic key generation algorithm" and "and specified cryptographic key sizes *[assignment: cryptographic key sizes*]" were deleted for clarity and better flow of the requirement. The parameters for generating asymmetric keys can be generated by using different criteria. By deleting the CC words, the element better states the intended requirement.

FCS_CKM.1.1(2) - **Refinement:** The TSF shall generate **asymmetric** cryptographic keys in accordance with a ~~specified cryptographic key generation algorithm~~ **domain parameter generator** and ***[selection:***

(4) a random number generator, and/or

(5) a prime number generator]

~~and specified cryptographic key sizes [*assignment: cryptographic key sizes*]~~ that meet the…

7 A deletion of CC text was performed in FCS_CKM.4.1. Rationale: The words "specified" and the assignment "*[assignment: cryptographic key destruction method]*" were deleted because FIPS PUB 140-2 does not provide specific names for the key destruction (zeroization) method.

FCS_CKM.4.1: **Refinement:** The TSF shall destroy cryptographic keys in accordance with a ~~specified~~ **cryptographic key destruction method** ~~[assignment: *cryptographic key destruction method*]~~ that meets …

8 A deletion of CC text was performed in FCS_COP.1.1(1). Rationale: The words "a specified cryptographic algorithm" and "and cryptographic key sizes *[assignment: cryptographic key sizes]*" were deleted for clarity and better flow of the requirement. The assignment was replaced with a selection that incorporates the algorithm and the key size for the corresponding algorithm.

FCS_COP.1.1(1) Refinement: The TSF shall perform **data encryption/decryption services** in accordance with ~~a specified cryptographic algorithm~~ a **NIST-approved implementation of the Advanced Encryption Standard (AES) cryptographic algorithm used in NIST-approved modes of operation and cryptographic key size of at least 128 bits** ~~and cryptographic key sizes *[assignment: cryptographic key sizes]*~~ that meets …

9 A deletion of CC text was performed in FCS_COP.1.1(2). Rationale: The words "a specified cryptographic signature algorithm" and "and cryptographic key sizes *[assignment: cryptographic key sizes]*" were deleted for clarity and better flow of the requirement. The assignment was replaced with a selection that incorporates the algorithm and the key size for the corresponding algorithm.

FCS_COP.1.1(2) - **Refinement**: The TSF shall perform **cryptographic signature services** in accordance with ~~a specified cryptographic algorithm~~ a **NIST-approved implementation of *[selection:***

(6) Digital Signature Algorithm (DSA) with a key size (modulus) of 2048 bits or greater, or

(7) RSA Digital Signature Algorithm (rDSA) with a key size (modulus) of 2048 bits or greater, or

(8) Elliptic Curve Digital Signature Algorithm (ECDSA) with a key size of 256 bits or greater]

~~and cryptographic key sizes *[assignment: cryptographic key sizes]*~~ that meets …

10 A deletion of CC text was performed in FCS_COP.1.1(4). Rationale: The words "a specified cryptographic" and "and cryptographic key sizes *[assignment: cryptographic key sizes]*" were deleted for clarity and better flow of the requirement. The assignment was replaced with a selection that incorporates the algorithm and the key size for the corresponding algorithm.

FCS_COP.1.1(4) - **Refinement:** The TSF shall perform **cryptographic key agreement services** in accordance with ~~a specified cryptographic algorithm~~ **a NIST-approved implementation of a *[selection:***

(1) Finite Field-based key agreement algorithm and cryptographic key sizes (modulus) of 2048 bits or greater

(2) Elliptic Curve-based key agreement algorithm and cryptographic key size of 256 bits or greater]

~~and cryptographic key sizes *[assignment: cryptographic key sizes]*~~ that meets …

11 A deletion of CC text was performed in FDP_ACC.2.1. Rationale: The words “subjects and objects covered by the SFP” were replaced with “them” at the end of the requirement to make requirement more clear and for better flow.

12 A deletion of CC text was performed in FDP_ACC.2.2. Rationale: The words “within the TSC” and “an access control SFP” were deleted because there is no need to specify that subjects and objects are within the TSC and to explicitly state the access control policy we are referring to (DAC).

FDP_ACC.2.2 **Refinement:** The TSF shall ensure that all operations between any subject ~~within the TSC~~ and any **named** object are covered by ~~an access control SFP~~ **the Discretionary Access Control policy**.

13 A deletion of CC text was performed in FDP_ACF.1.2. Rationale: The word “controlled” was deleted because there is no need to specify that subjects and objects are controlled.

FDP_ACF.1.2 **Refinement:** The TSF shall enforce the following rules to determine if an operation among ~~controlled~~ subjects and ~~controlled~~ named objects is …

14 A deletion of CC text was performed in FIA_UAU.6.1. Rationale: The words “under the conditions” were deleted for better clarity and flow on the element.

FIA_UAU.6.1 **Refinement:** The TSF shall re-authenticate the user ~~under the conditions~~ **when changing authentication data**.

15 A deletion of CC text was performed in FMT_MOF.1.1(2). Rationale: The words "the functions" were deleted for clarity and better flow of the requirement.

FMT_MOF.1.1(2) **Refinement:** The TSF shall restrict the ability to **manage** ~~the functions~~ **the values of security attributes associated with user authentication data** to **authorized administrators.**

16 A deletion of CC text was performed in FMT_MSA.1.1. Rationale: The assignment “[assignment: list of security attributes]" was deleted for clarity and better flow of the requirement. The requirement is intended to restrict any changes to any of the values of all object security attributes.

FMT_MSA.1.1 **Refinement:** The TSF shall enforce the **Discretionary Access Control policy** to restrict the ability to **change** the **value of the object** security attributes ~~[assignment: list of security attributes]~~ to **authorized administrators and owners of the object**.

17 A deletion of CC text was performed in FMT_MSA.1.1. Rationale: The assignment “[assignment: list of security attributes]" was deleted for clarity and better flow of the requirement. The requirement is intended to restrict any changes to any of the values of all object security attributes.

FMT_MSA.1.1 **Refinement:** The TSF shall enforce the **Discretionary Access Control policy** to restrict the ability to **change** the **value of the object** security attributes ~~[assignment: list of security attributes]~~ to **authorized administrators and owners of the object**.

18

FMT_MSA.2.1 Refinement: The TSF shall ensure that only ~~secure~~ valid values are accepted for security attributes.

19 A deletion of CC text was performed in FMT_MTD.1.1(6). Rationale: The words "restrict" and the assignment “to [assignment: the authorized identified roles].” were deleted for clarity and better flow of the requirement.

FMT_MTD.1.1(6) **Refinement:** The TSF shall **prevent** ~~restrict~~ ~~the ability to~~ **reading of authentication data**~~to~~ ~~[assignment: the authorized identified roles]~~.

20 A deletion of CC text was performed in FMT_REV.1.2 (1). Rationale: The word "rules" was deleted for clarity and better flow of the requirement.

FMT_REV.1.2(1) **Refinement:** The TSF shall enforce the ~~rules~~ **immediate revocation of security-relevant authorizations**.

21 A deletion of CC text was performed in FMT_REV.1.1 (2). Rationale: The words "associated with" were deleted for clarity and better flow of the requirement.

FMT_REV.1.1 (2) **Refinement:** The TSF shall restrict the ability to revoke security attributes ~~associated~~ **of named** objects within the TSC to **owners of the named object and authorized administrators.**

22 A deletion of CC text was performed in FMT_REV.1.2 (2). Rationale: The word "rules" was deleted for clarity and better flow of the requirement.

FMT_REV.1.2 (2) **Refinement:** The TSF shall enforce the ~~rules~~ **revocation of access rights associated with operating system controlled files when an access check is made**.

23 A deletion of CC text was performed in FMT_SAE.1.2. Rationale: The words " For each of these security attributes," and "for the indicated security attribute**"** were deleted for clarity and better flow of the requirement.

FMT_SAE.1.2 **Refinement:** ~~For each of these security attributes~~, The TSF shall be able to **lock out the associated authorized user account** after the expiration time ~~for the indicated security attribute~~ has passed.

24 A deletion of CC text was performed in FMT_SMR.2.3. Rationale: The words "the conditions" and "are satisfied" were deleted for clarity and better flow of the requirement.

FMT_SMR.2.3 Refinement: The TSF shall ensure that ~~the conditions~~ roles are distinct and that no overlap of allowed operations exists between roles ~~are satisfied~~.

25 The CC text "for the following entities:" was deleted in FPT_RPL.1.1. Rationale: clarity and better flow of the requirement

FPT_RPL.1.1 **Refinement:** The TSF shall **be able to** detect replay ~~for the following entities:~~ **of TSF data transmitted between separate parts of the TOE through the use of the TSF-provided cryptographic services.**

26 The CC text was reworded in FPT_RPL.1.2 Rationale: The definition was changed from:

FPT_RPL.1.2 The TSF shall perform [assignment: list of specific actions] when replay is detected.

to the following definition:

FPT_RPL.1.2 Refinement: Upon detection of **TSF data** replay, the TSF shall take the following actions…

27 A deletion of CC text was performed in FPT_SEP.2.3. Rationale: The words "security domain" were deleted to scope the requirement an address space. The words "their", "them", and "those SPFs" were deleted for grammatical reasons since this element refers to cryptography and not SPFs.

FPT_SEP.2.3 **Refinement:** The TSF shall maintain **separation of** the part of the TSF related to **cryptography** ~~in a security domain for their own execution~~ that protects ~~them~~ **it** from interference and tampering by the remainder of the TSF and by subjects untrusted with respect ~~to those SPFs~~ **cryptography**.

28 A deletion of CC text was performed in FPT_TST.1.1(2). Rationale: The word "TSF" was deleted to allow for the demonstration of the correct operation of a number of cryptographic related self tests.

FPT_TST.1.1(2) **Refinement:** The TSF shall run a suite of self-**tests in accordance with FIPS PUB 140-2, Level 4 (as identified in Table 5.3)** during initial start-up **(on power on)**, at the request of the **cryptographic administrator (on demand)**, **under various conditions,** and periodically **(at least once a day)** to demonstrate the correct operation of the ~~TSF~~ **following …**

29 A deletion of CC text was performed in FPT_TST.1.2(2). Rationale: The word "users" was deleted to replace it with the role of " cryptographic administrator". "Only authorized cryptographic administrators should be given the capability to verify the integrity of cryptographically related TSF data.

FPT_TST.1.2 (2) **Refinement:** The TSF shall provide authorized ~~users~~ **cryptographic administrators** with the capability to verify the integrity of **cryptographically related** TSF data.

30 A deletion of CC text was performed in FPT_TST.1.3(2). Rationale: The word “users” was deleted to replace it with the role of " cryptographic administrator". Only authorized cryptographic administrators should be given the capability to verify the integrity of cryptographically related TSF executable code.

FPT_TST.1.3(2) **Refinement:** The TSF shall provide authorized ~~users~~ **cryptographic administrators** with the capability to verify the integrity of stored **cryptographically related** TSF executable code.

31 A deletion of CC text was performed in FPT_TST.1.1(3). Rationale: The word "the TSF" was deleted to allow for the demonstration of the correct operation of each key generation component and the word “perform” replaced “run a suite of” for clarity and better flow of the requirement.

FPT_TST.1.1(3) **Refinement:** The TSF shall ~~run a suite of~~ perform self-tests **immediately after generation of a key** to demonstrate the correct operation of ~~the TSF~~ **each key generation component**. **If any of these tests fails, that generated key shall not be used, the cryptographic module shall react as required by FIPS PUB 140 for failing a self-test, and this event will be audited.**

32 A deletion of CC text was performed in FPT_TST.1.2(2). Rationale: The word "users" was deleted to replace it with the role of "cryptographic administrator".

FPT_TST.1.2(2) **Refinement:** The TSF shall provide authorized ~~users~~ **cryptographic administrators** with the capability to verify the integrity of TSF data **related to the key generation**.

33 A deletion of CC text was performed in FPT_TST.1.3(2). Rationale: The word “users” was deleted to replace it with the role of "cryptographic administrator".

FPT_TST.1.3(2) **Refinement:** The TSF shall provide authorized ~~users~~ **cryptographic administrators** with the capability to verify the integrity of stored TSF executable code **related to the key generation**.

34 A deletion of CC text was performed in FTA_LSA.1.1. Rationale: The words "the session security attributes" were deleted for clarity and better flow of the requirement.

FTA_LSA.1.1 **Refinement**: The TSF shall restrict the scope of ~~the session security attributes~~ **roles and user privileges** based on **location, time, and day**.

35 A deletion of CC text was performed in FTA_MCS.1.1. Rationale: The words "restrict the" and “that belong to the same” were deleted for clarity and better flow of the requirement.

FTA_MCS.1.1 **Refinement:** The TSF shall ~~restrict the~~ enforce a maximum number of concurrent **interactive** sessions ~~that belong to the same~~ per user.

36 A deletion of CC text was performed in FTA_MCS.1.2. Rationale: The words "enforce, by default, a limit of" were deleted to refine the requirement to allow for a settable limit of sessions per user.

FTA_MCS.1.2 **Refinement:** The TSF shall ~~enforce, by default, a limit of~~ allow **an administrator to set the maximum number of concurrent interactive** sessions per user.

37 A deletion of CC text was performed in FTA_SSL.1.2. Rationale: The words "following events to occur” were

deleted for clarity and better flow of the requirement.

FTA_SSL.1.2 **Refinement:** The TSF shall require the ~~following events to occur~~ **user to re-authenticate** ~~prior~~ to unlock~~ing~~ the session.

38 A deletion of CC text was performed in FTA_SSL.2.2. Rationale: The words "following events to occur" were deleted for clarity and better flow of the requirement.

FTA_SSL.2.2 **Refinement:** The TSF shall require the ~~following events to occur~~ **user to re-authenticate** ~~prior~~ to unlock~~ing~~ the session.

6. Security Assurance Requirements

80 This section contains the detailed security assurance requirements for operating systems supporting single-level and system high systems in environments requiring medium robustness. The requirements contained in this section are either selected from Part 3 of the CC or have been explicitly stated (with short names ending in "_EXP"). Table 6.1 lists the explicitly stated assurance components.

Table 6.1 - Explicit Assurance Requirements

Explicit Component	Component Behavior Name
ADV_ARC_EXP.1	Architectural Design
ADV_FSP_EXP.1	Functional specification with Complete Summary
ADV_HLD_EXP.1	Security Enforcing High-Level Design
ADV_IMP_EXP.2	Implementation Representation
ADV_INT_EXP.1	Modular Decomposition
ADV_LLD_EXP.1	Security-Enforcing Low-Level Design
AMA_AMP_EXP.1	Assurance Maintenance Plan
ATE_COV_EXP.2	Analysis of Coverage
AVA_CCA_EXP.1	Cryptographic Module Covert Channel Analysis

81 The combination of assurance components is equivalent to an Evaluated Assurance Level 4 with augmentation (EAL4+).[19] The augmented assurances required are in the areas of vulnerability analysis/penetration testing, development, and covert channel analysis for cryptography. The intended TOE environment and the value of information processed by this environment establish the need for the TOE to be evaluated at this EAL level[20]. These security assurance requirements are summarized in Table 6.2. Note that flaw remediation (ALC_FLR) and maintenance of assurance (AMA_AMP_EXP) have also been chosen even though the CC does not assign these components to a specific EAL level.

[19] The assurance components are "equivalent to EAL4" and not EAL4 as written in the CC because some EAL4 components have been modified (and made explicit) to reflect EAL4 as it is commonly practiced in mutually accepted evaluations to date. These are denoted in Table 6.2 under the EAL4 column with an asterisk (*).

[20] Refer to the "Mutual Recognition of Common Criteria Certificates" section 1.3 to read conditions for the CC certificate to be mutually recognized for PPs with EALs higher than 4.

Table 6.2 - Summary of Assurance Components by Evaluation Assurance Level

Assurance Class	Assurance Family	Assurance Components by Evaluation Assurance Level						
		EAL1	EAL2	EAL3	**EAL4**	EAL5	EAL6	EAL7
Configuration Management	ACM_AUT				**1**	1	2	2
	ACM_CAP	1	2	3	**4**	4	5	5
	ACM_SCP			1	**2**	3	3	3
Delivery and Operation	ADO_DEL		1	1	**2**	2	2	3
	ADO_IGS	1	1	1	**1**	1	1	1
Development	ADV_ARC_EXP				**(1)**			
	ADV_FSP_EXP	1	1	1	**2***	3	3	4
	ADV_HLD_EXP		1	2	**2***	3	4	5
	ADV_IMP_EXP				1	**2***	3	3
	ADV_INT_EXP					**1****	2	3
	ADV_LLD_EXP				**1***	1	2	2
	ADV_RCR	1	1	1	**1**	2	2	3
	ADV_SPM				**1**	3	3	3
Guidance Documents	AGD_ADM	1	1	1	**1**	1	1	1
	AGD_USR	1	1	1	**1**	1	1	1
Life cycle Support	ALC_DVS			1	**1**	1	2	2
	ALC_FLR				**(2)**			
	ALC_LCD				**1**	2	2	3
	ALC_TAT				**1**	2	3	3
Maintenance of Assurance	AMA_AMP_EXP				**(1)**			
Tests	ATE_COV_EXP		1	2	**2***	2	3	3
	ATE_DPT			1	1	**2**	2	3
	ATE_FUN		1	1	**1**	1	2	2
	ATE_IND	1	2	2	**2**	2	2	3
Vulnerability Assessment	AVA_CCA_EXP**					**1**	**2*****	2
	AVA_MSU			1	**2**	2	3	3
	AVA_SOF		1	1	**1**	1	1	1
	AVA_VLA		1	1	2	**3**	4	4

***** Explicit components that are equivalent to CC EAL4 components.

** The Modular Decomposition component (ADV_INT_EXP) has been modified to reflect medium robustness by levying modularity requirements on security-enforcing entities within the TSF.

*** The covert channel analysis is performed only upon the cryptographic module.

6.1 Configuration Management (ACM)

6.1.1 CM Automation (ACM_AUT)

6.1.1.1 Partial CM Automation (ACM_AUT.1)

ACM_AUT.1.1D The developer shall use a CM system.

ACM_AUT.1.2D The developer shall provide a CM plan.

ACM_AUT.1.1C The CM system shall provide an automated means by which only authorized changes are made to the TOE implementation

representation.

ACM_AUT.1.2C The CM system shall provide an automated means to support the generation of the TOE.

ACM_AUT.1.3C The CM plan shall describe the automated tools used in the CM system.

ACM_AUT.1.4C The CM plan shall describe how the automated tools are used in the CM system.

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

6.1.2 CM Capabilities (ACM_CAP)

6.1.2.1 Generation Support and Acceptance Procedures (ACM_CAP.4)

ACM_CAP.4.1D The developer shall provide a reference for the TOE.

ACM_CAP.4.2D The developer shall use a CM system.

ACM_CAP.4.3D The developer shall provide CM documentation.

ACM_CAP.4.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.4.2C The TOE shall be labeled with its reference.

ACM_CAP.4.3C The CM documentation shall include a configuration list, a CM plan, and an acceptance plan.

ACM_CAP.4.4C The configuration list shall uniquely identify all configuration items that comprise the TOE.

ACM_CAP.4.5C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.4.6C The CM documentation shall describe the method used to uniquely identify the configuration items that comprise the TOE.

ACM_CAP.4.7C The CM system shall uniquely identify all configuration items that comprise the TOE.

ACM_CAP.4. 8C The CM plan shall describe how the CM system is used.

ACM_CAP.4.9C The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.4.10C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.4.11C The CM system shall provide measures such that only authorized changes are made to the configuration items.

ACM_CAP.4.12C The CM system shall support the generation of the TOE.

ACM_CAP.4.13C The acceptance plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.

ACM_CAP.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

6.1.3 CM Scope (ACM_SCP)

6.1.3.1 Problem Tracking CM Coverage (ACM_SCP.2)

ACM_SCP.2.1D The developer shall provide a list of configuration items for the TOE.

ACM_SCP.2.1C The list of configuration items shall include the following: the TOE implementation representation, security flaws; and the evaluation evidence required by the assurance components in the ST.

ACM_SCP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

6.2 Delivery and Operation (ADO)

6.2.1 Delivery (ADO_DEL)

6.2.1.1 Detection of Modification (ADO_DEL.2)

ADO_DEL.2.1D The developer shall document procedures for delivery of the TOE or parts of it to the user.

ADO_DEL.2.2D The developer shall use the delivery procedures.

ADO_DEL.2.1C The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.

ADO_DEL.2.2C The delivery documentation shall describe how the various procedures and technical measures provide for the detection of modifications, or any discrepancy between the developer's master copy and the version received at the user site.

ADO_DEL.2.3C The delivery documentation shall describe how the various procedures allow detection of attempts to masquerade as the developer, even in cases in which the developer has sent nothing to the user's site.

ADO_DEL.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

6.2.2 Installation, Generation and Start-up (ADO_IGS)

6.2.2.1 Installation, Generation, and Start-Up Procedures (ADO_IGS.1)

ADO_IGS.1.1D The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

ADO_IGS.1.1C The installation, generation and start-up documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.

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

ADO_IGS.1.2E The evaluator shall determine that the installation, generation, and start-up procedures result in a secure configuration.

6.3 Development Documentation (ADV)

6.3.1 Architectural Design (ADV_ARC)

6.3.1.1 Explicit: Architectural Design (ADV_ARC_EXP.1)

ADV_ARC_EXP.1.1D The developer shall provide the architectural design of the TSF.

ADV_ARC_EXP.1.1C The architectural design shall be at a level of detail commensurate with the description of the TSF as described in the TOE design document.

ADV_ARC_EXP.1.2C The architectural design shall be internally consistent.

ADV_ARC_EXP.1.3C The architectural design shall describe the design of the TSF self-protection mechanisms.

ADV_ARC_EXP.1.4C The architectural design shall describe the design of the TSF in detail sufficient to determine that the security enforcing mechanisms cannot be bypassed.

ADV_ARC_EXP.1.5C The architectural design shall justify that the design of the TSF achieves the self-protection function.

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

ADV_ARC_EXP.1.2E The evaluator shall analyze the architectural design and dependent documentation to determine that FPT_SEP and FPT_RVM are accurately implemented in the TSF.

6.3.2 Functional Specification (ADV_FSP)

6.3.2.1 Explicit: Functional Specification with Complete Summary (ADV_FSP_EXP.1)

ADV_FSP_EXP.1.1D The developer shall provide a functional specification.

ADV_FSP_EXP.1.1C The functional specification shall completely represent the TSF.

ADV_FSP_EXP.1.2C The functional specification shall be internally consistent.

ADV_FSP_EXP.1.3C The functional specification shall describe the external TSF interfaces (TSFIs).

ADV_FSP_EXP.1.4C The functional specification shall designate each external TSFI as security enforcing or security supporting.

ADV_FSP_EXP.1.5C The functional specification shall describe the purpose and method of use for each external TSFI.

ADV_FSP_EXP.1.6C The functional specification shall identify and describe all parameters associated with each external TSFI.

ADV_FSP_EXP.1.7C For security enforcing external TSFIs, the functional specification shall describe the security enforcing effects and security enforcing exceptions.

ADV_FSP_EXP.1.8C For security enforcing external TSFIs, the functional specification shall describe direct error messages resulting from security enforcing effects and exceptions.

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

ADV_FSP_EXP.1.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the user-visible TOE security functional requirements.

6.3.3 High-Level Design (ADV_HLD)

6.3.3.1 Explicit: Security Enforcing High-Level Design (ADV_HLD_EXP.1)

ADV_HLD_EXP.1.1D The developer shall provide the high-level design of the TOE.

ADV_HLD_EXP.1.1C The high-level design shall describe the structure of the TOE in terms of subsystems.

ADV_HLD_EXP.1.2C The high-level design shall be internally consistent.

ADV_HLD_EXP.1.3C The high-level design shall describe the design of the TOE in sufficient detail to determine what subsystems of the TOE are part of the TSF.

ADV_HLD_EXP.1.4C The high-level design shall describe the design of the TOE in sufficient detail to determine what subsystems of the TOE are part of the TSF.

ADV_HLD_EXP.1.5C The high-level design shall identify all subsystems in the TSF, and designate them as either security enforcing or security supporting.

ADV_HLD_EXP.1.6C The high-level design shall describe the structure of the security-enforcing subsystems.

ADV_HLD_EXP.1.7C For security-enforcing subsystems, the high-level design shall describe the design of the security-enforcing behavior.

ADV_HLD_EXP.1.8C For security-enforcing subsystems, the high-level design shall summarize any non-security-enforcing behavior.

ADV_HLD_EXP.1.9C The high-level design shall summarize the behavior for security-supporting subsystems.

ADV_HLD_EXP.1.10C The high-level design shall summarize all interactions between subsystems of the TSF.

ADV_HLD_EXP.1.11C The high-level design shall describe any interactions between the security-enforcing subsystems of the TSF.

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

ADV_HLD_EXP.1.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of all user-visible TOE security functional requirements with the exception of FPT_SEP and FPT_RVM.

6.3.4 Implementation Representation (ADV_IMP_EXP)

6.3.4.1 Explicit: Implementation of the TSF (ADV_IMP_EXP.2)

ADV_IMP_EXP.2.1D The developer shall provide the implementation representation for the entire TSF.

ADV_IMP_EXP.2.1C The implementation representation shall unambiguously define the TSF to a level of detail such that an operational TSF can be produced without further design decisions.

Application Note: The implementation representation includes source code and processor programming manuals.

ADV_IMP_EXP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP_EXP.2.2E The evaluator shall determine that the implementation representation is an accurate and complete instantiation of the TOE security functional requirements.

6.3.5 TSF Internals (ADV_INT)

6.3.5.1 Explicit: Modular Decomposition (ADV_INT_EXP.1)

ADV_INT_EXP.1.1D The developer shall design and implement the TSF using modular decomposition.

ADV_INT_EXP.1.2D The developer shall use sound software engineering principles to achieve the modular decomposition of the TSF.

ADV_INT_EXP.1.3D The developer shall design software modules such that they exhibit good internal structure and are not overly complex.

ADV_INT_EXP.1.4D The developer shall design software modules that implement the DAC, MAC, and MIC policies and cryptography such that they exhibit only functional, sequential, communicational, or temporal cohesion, with limited exceptions.

ADV_INT_EXP.1.5D The developer shall design the SFP-enforcing software modules such that they exhibit only call or common coupling, with limited exceptions.

Application Note: SFP-enforcing software modules are TSF modules that implement a specific SFP identified in ADV_INT_EXP.1.4D.

ADV_INT_EXP.1.6D The developer shall implement TSF software modules using coding standards that result in good internal structure that is not overly complex.

ADV_INT_EXP.1.7D The developer shall provide a software architectural description.

ADV_INT_EXP.1.1C The software architectural description shall identify the SFP-enforcing and non-SFP-enforcing modules.

ADV_INT_EXP.1.2C The TSF software modules shall be identical to those described by the low level design (ADV_LLD_EXP.1.4C).

ADV_INT_EXP.1.3C The software architectural description shall provide a justification for the designation of non-SFP-enforcing modules that interact with the SFP-enforcing module(s).

ADV_INT_EXP.1.4C The software architectural description shall describe the process used for modular decomposition.

ADV_INT_EXP.1.5C The software architectural description shall describe how the TSF design is a reflection of the modular decomposition process.

ADV_INT_EXP.1.6C The software architectural description shall include the coding standards used in the development of the TSF.

ADV_INT_EXP.1.7C The software architectural description shall provide a justification, on a per module basis, of any deviations from the coding standards.

ADV_INT_EXP.1.8C The software architectural description shall include a coupling analysis that describes intermodule coupling for the SFP-enforcing modules.

ADV_INT_EXP.1.9C The software architectural description shall provide a justification, on a per module basis, for any coupling or cohesion exhibited by SFP-enforcing modules, other than those permitted.

ADV_INT_EXP.1.10C The software architectural description shall provide a justification, on a per module basis, that the SFP-enforcing modules are not overly complex.

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

ADV_INT_EXP.1.2E The evaluator shall perform a cohesion analysis for the modules that substantiates the type of cohesion claimed for a subset of SFP-enforcing modules.

ADV_INT_EXP.1.3E The evaluator shall perform a complexity analysis for a subset of TSF modules.

6.3.6 Low-level Design (ADV_LLD)

6.3.6.1 Explicit: Security-Enforcing Low-Level Design (ADV_LLD_EXP.1)

ADV_LLD_EXP.1.1D The developer shall provide the low-level design of the TSF.

ADV_LLD_EXP.1.1C The presentation of the low-level design shall be separate from the implementation representation.

ADV_LLD_EXP.1.2C The low-level design shall be internally consistent.

ADV_LLD_EXP.1.3C The low-level design shall describe the TSF in terms of modules, designating each module as either security-enforcing or security-supporting.

ADV_LLD_EXP.1.4C The low-level design shall identify and describe data that are common to more than one module, where any of the modules is a security-enforcing module.

ADV_LLD_EXP.1.5C The low level design shall describe each security-enforcing module in terms of its purpose, interfaces, return values, called interfaces to other modules, and global variables.

ADV_LLD_EXP.1.6C For each security-enforcing module, the low level design shall provide an algorithmic description detailed enough to represent the TSF implementation.

Application Note: An algorithmic description contains sufficient detail such that two different programmers would produce functionally equivalent code, although data structures, programming methods, etc. may differ.

ADV_LLD_EXP.1.7C The low level design shall describe each security-supporting module in terms of its purpose and interaction with other modules.

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

ADV_LLD_EXP.1.2E The evaluator shall determine that the low-level design is an accurate and complete instantiation of all TOE security functional requirements, with the exception of FPT_SEP and FPT_RVM.

6.3.7 Representation Correspondence (ADV_RCR)

6.3.7.1 Informal Correspondence Demonstration (ADV_RCR.1)

ADV_RCR.1.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

ADV_RCR.1.1C For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

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

6.3.8 Security Policy Modeling (ADV_SPM)

6.3.8.1 Informal TOE Security Policy Model (ADV_SPM.1)

ADV_SPM.1.1D The developer shall provide a TSP model.

ADV_SPM.1.2D The developer shall demonstrate correspondence between the functional specification and the TSP model.

ADV_SPM.1.1C The TSP model shall be informal.

ADV_SPM.1.2C The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

Application Note: Security policies that can be modeled include descriptions of at least the following security policies: Identification and Authentication, Discretionary Access Control, Audit, and Cryptography.

ADV_SPM.1.3C The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.1.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model.

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

6.4 Guidance Documents (AGD)

6.4.1 Administrator Guidance (AGD_ADM)

6.4.1.1 Administrator Guidance (AGD_ADM.1)

AGD_ADM.1.1D The developer shall provide administrator guidance addressed to system administrative personnel.

AGD_ADM.1.1C The administrator guidance shall describe the administrative functions and interfaces available to the administrator of the TOE.

Application Note: Administrators of the TOE include the "authorized administrator" and

"cryptographic administrator" roles (see FMT_SMR.2).

AGD_ADM.1.2C The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.3C The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.4C The administrator guidance shall describe all assumptions regarding user behavior that are relevant to secure operation of the TOE.

AGD_ADM.1.5C The administrator guidance shall describe all security parameters under the control of the administrator, indicating secure values as appropriate.

AGD_ADM.1.6C The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_ADM.1.7C The administrator guidance shall be consistent with all other documentation supplied for evaluation.

AGD_ADM.1.8C The administrator guidance shall describe all security requirements for the IT environment that are relevant to the administrator.

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

6.4.2 User Guidance (AGD_USR)

6.4.2.1 User Guidance (AGD_USR.1)

AGD_USR.1.1D The developer shall provide user guidance.

AGD_USR.1.1C The user guidance shall describe the functions and interfaces available to the non-administrative users of the TOE.

Application Note: This includes guidance for the users of the cryptographic module.

AGD_USR.1.2C The user guidance shall describe the use of user-accessible security functions provided by the TOE.

AGD_USR.1.3C The user guidance shall contain warnings about user-accessible functions and privileges that should be controlled in a secure processing environment.

AGD_USR.1.4C The user guidance shall clearly present all user

responsibilities necessary for secure operation of the TOE, including those related to assumptions regarding user behavior found in the statement of TOE security environment.

AGD_USR.1.5C The user guidance shall be consistent with all other documentation supplied for evaluation.

AGD_USR.1.6C The user guidance shall describe all security requirements for the IT environment that are relevant to the user.

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

6.5 Life Cycle Support (ALC)

6.5.1 Development Security (ALC_DVS)

6.5.1.1 Identification of Security Measures (ALC_DVS.1)

ALC_DVS.1.1D The developer shall produce development security documentation.

ALC_DVS.1.1C The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.

ALC_DVS.1.2C The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

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

ALC_DVS.1.2E The evaluator shall confirm that the security measures are being applied.

6.5.2 Flaw Remediation (ALC_FLR)

6.5.2.1 Flaw Reporting Procedures (ALC_FLR.2)

ALC_FLR.2.1D The developer shall provide flaw remediation procedures addressed to TOE developers.

ALC_FLR.2.2D The developer shall establish a procedure for accepting and acting upon user reports of security flaws and requests for corrections to those flaws.

ALC_FLR.2.3D The developer shall provide flaw remediation guidance addressed to TOE users.

ALC_FLR.2.1C The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.2.2C The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

ALC_FLR.2.3C The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.2.4C The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

ALC_FLR.2.5C The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.

ALC_FLR.2.6C The procedures for processing reported security flaws shall ensure that any reported flaws are corrected and the correction issued to TOE users.

ALC_FLR.2.7C The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.

ALC_FLR.2.8C The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.

ALC_FLR.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

6.5.3 Life Cycle Definition (ALC_LCD)

6.5.3.1 Developer Defined Life-Cycle Model (ALC_LCD.1)

ALC_LCD.1.1D The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

ALC_LCD.1.2D The developer shall provide life-cycle definition documentation.

ALC_LCD.1.1C The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.

ALC_LCD.1.2C The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.

ALC_LCD.1.1E The evaluator shall confirm that the information provided

meets all requirements for content and presentation of evidence.

6.5.4 Tools and Techniques (ALC_TAT)

6.5.4.1 Well-Defined Development Tools (ALC_TAT.1)

ALC_TAT.1.1D The developer shall identify the development tools being used for the TOE.

ALC_TAT.1.2D The developer shall document the selected implementation-dependent options of the development tools.

ALC_TAT.1.1C All development tools used for implementation shall be well defined.

Application Note: The development tools include the compiler used to generate the TOE.

ALC_TAT.1.2C The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

ALC_TAT.1.3C The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.

Application Note: This documentation includes the compiler options used during the generation of the TOE.

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

6.6 Ratings Maintenance (AMA)

6.6.1 Assurance Maintenance Plan (AMA_AMP)

6.6.1.1 Explicit: Assurance Maintenance Plan (AMA_AMP_EXP.1)

AMA_AMP_EXP.1.1D - The developer shall provide an Assurance Maintenance (AM) Plan.

AMA_AMP_EXP.1.1C - The AM Plan shall identify the assurance baseline.

AMA_AMP_EXP.1.2C - The AM Plan shall contain or reference a brief description of the TOE, including the security functionality it provides.

AMA_AMP_EXP.1.3C - The AM Plan shall characterize the types of changes to the assurance baseline that are covered by the plan.

AMA_AMP_EXP.1.4C - The AM Plan shall describe the planned Target of Maintenance (TOM) release-cycle.

AMA_AMP_EXP.1.5C - The AM Plan shall identify the planned schedule of AM audits and the conditions for the end of maintenance.

AMA_AMP_EXP.1.5C - The AM Plan shall justify the planned schedule of AM audits and the conditions for the end of maintenance.

AMA_AMP_EXP.1.6C - The AM Plan shall identify the processes that are necessary for assigning, and ensuring currency of knowledge of, individual(s) assuming the role of security analyst.

AMA_AMP_EXP.1.7C - The AM Plan shall define the relationship between the security analyst and the development of the evidence.

AMA_AMP_EXP.1.8C - The AM Plan shall identify the qualifications that are necessary for the individual(s) identified as the security analyst.

AMA_AMP_EXP.1.9C - The AM Plan shall describe the procedures by which changes to the assurance baseline will be identified.

AMA_AMP_EXP.1.10C - The AM Plan shall describe the procedures that are necessary to be applied to the TOM to maintain the assurance established for the certified TOE.

AMA_AMP_EXP.1.11C - The AM Plan shall describe the controls and mechanisms that are necessary to ensure that the procedures documented in the AM Plan are followed.

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

6.7 Testing (ATE)

6.7.1 Coverage (ATE_COV)

6.7.1.1 Analysis of Coverage (ATE_COV_EXP.2)

ATE_COV_EXP.2.1D The developer shall provide an analysis of the test coverage.

ATE_COV_EXP.2.1C The analysis of the test coverage shall demonstrate the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.

ATE_COV_EXP.2.2C The analysis of the test coverage shall demonstrate that the correspondence between the TSF as described in the functional specification and the tests identified in the test documentation is complete.

ATE_COV_EXP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_COV_EXP.2.2E For cryptographic portions of the TOE, an NSA evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

6.7.2 Depth (ATE_DPT)

6.7.2.1 Testing: Low-Level Design (ATE_DPT.2)

ATE_DPT.2.1D The developer shall provide the analysis of the depth of testing.

ATE_DPT.2.1C The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design and low-level design.

ATE_DPT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

6.7.3 Functional Tests (ATE_FUN)

6.7.3.1 Functional testing (ATE_FUN.1)

ATE_FUN.1.1D **Refinement:** The developer shall test the TSF **including stress testing the boundary conditions of all external interfaces** and document the results.

Application Note: Stress testing of boundary conditions must be provided for all external TSF interfaces. However, the testing is not expected to be, nor would it be feasible to be, exhaustive. The test documentation should describe the philosophy of the approach to test the interface boundary conditions and should present evidence that the approach is sufficient.

ATE_FUN.1.2D The developer shall provide test documentation.

ATE_FUN.1.1C The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.

ATE_FUN.1.2C The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.1.3C The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests.

ATE_FUN.1.4C The expected test results shall show the anticipated outputs

from a successful execution of the tests.

ATE_FUN.1.5C The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.

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

6.7.4 Independent Testing (ATE_IND)

6.7.4.1 Independent Testing - Sample (ATE_IND.2)

ATE_IND.2.1D The developer shall provide the TOE for testing.

ATE_IND.2.1C The TOE shall be suitable for testing.

ATE_IND.2.2C The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

ATE_IND.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2.2E The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.

ATE_IND.2.3E The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

6.8 Vulnerability Assessment (AVA)

6.8.1 Explicit: Cryptographic Module Covert Channel Analysis (AVA_CCA_EXP)

6.8.1.1 Explicit: Systematic Cryptographic Module Covert Channel Analysis (AVA_CCA_EXP.2)

Application Note: The covert channel analysis is performed only upon the cryptographic module; a search is made for the leakage of critical cryptographic security parameters from the cryptographic module, rather than a violation of an information control policy. Inappropriate handling / leakage of any critical cryptographic security parameters (covered or not) that by design and implementation lie outside the cryptographic module is not addressed by this CCA. Thus, leakage of such parameters in such designs and implementations must be investigated by other means.

AVA_CCA_EXP.2.1D For the cryptographic module, the developer shall conduct a search for covert channels for the leakage of critical cryptographic security parameters whose disclosure would compromise the security provided by the module.

Application Note: The remainder of the TOE need not be subjected to a covert channel analysis. (Ideally, a covert channel analysis on the entire TSF would determine if TSF interfaces can be used covertly for the leakage of critical cryptographic security parameters. While such extensive covert channel analysis is more complete, it is also difficult and expensive. At this time it is considered beyond the scope of effort and cost considered reasonable for COTS medium robustness products. Consequently, covert channel analysis has been limited here to the cryptographic module, but that analysis limitation does come with some added risk of unknown leakage from other parts of the TOE.)

AVA_CCA_EXP.2.2D The developer shall provide covert channel analysis documentation.

AVA_CCA_EXP.2.1C The analysis documentation shall identify covert channels in the cryptographic module and estimate their capacity.

AVA_CCA_EXP.2.2C The analysis documentation shall describe the procedures used for determining the existence of covert channels in the cryptographic module, and the information needed to carry out the covert channel analysis.

AVA_CCA_EXP.2.3C The analysis documentation shall describe all assumptions made during the covert channel analysis.

AVA_CCA_EXP.2.4C The analysis documentation shall describe the method used for estimating channel capacity, based on worst case scenarios.

AVA_CCA_EXP.2.5C The analysis documentation shall describe the worst case exploitation scenario for each identified covert channel.

AVA_CCA_EXP.2.6C The analysis documentation shall provide evidence that the method used to identify covert channels is systematic.

AVA_CCA_EXP.2.1E The NSA evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_CCA_EXP.2.2E The NSA evaluator shall confirm that the results of the covert channel analysis show that the cryptographic module meets its functional requirements.

AVA_CCA_EXP.2.3E The NSA evaluator shall selectively validate the covert channel analysis through independent analysis and testing.

6.8.2 Misuse (AVA_MSU)

6.8.2.1 Validation of Analysis (AVA_MSU.2)

AVA_MSU.2.1D The developer shall provide guidance documentation.

AVA_MSU.2.2D The developer shall document an analysis of the guidance

documentation.

AVA_MSU.2.1C The guidance documentation 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.

AVA_MSU.2.2C The guidance documentation shall be complete, clear, consistent and reasonable.

AVA_MSU.2.3C The guidance documentation shall list all assumptions about the intended environment.

AVA_MSU.2.4C The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).

AVA_MSU.2.5C The analysis documentation shall demonstrate that the guidance documentation is complete.

AVA_MSU.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.2.2E The evaluator shall repeat all configuration and installation procedures, and other procedures selectively, to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.

AVA_MSU.2.3E The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.

AVA_MSU.2.4E The evaluator shall confirm that the analysis documentation shows that guidance is provided for secure operation in all modes of operation of the TOE.

6.8.3 Strength of TOE security functions (AVA_SOF)

6.8.3.1 Strength of TOE Security Function Evaluation (AVA_SOF.1)

Application Note: The security functions, for which strength of function claims are made, are identified in sections 5.2.2 and 5.4.3.

AVA_SOF.1.1D The developer shall perform a strength of TOE security function analysis for each mechanism identified in the ST as having a strength of TOE security function claim.

AVA_SOF.1.1C For each mechanism with a strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the minimum strength level defined in the PP/ST.

AVA_SOF.1.2C For each mechanism with a specific strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the specific strength of function metric defined in the PP/ST.

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

AVA_SOF.1.2E The evaluator shall confirm that the strength claims are correct.

6.8.4 Vulnerability Analysis (AVA_VLA)

6.8.4.1 Moderately Resistant (AVA_VLA.3)

AVA_VLA.3.1D The developer shall perform a vulnerability analysis.

AVA_VLA.3.2D The developer shall provide vulnerability analysis documentation.

AVA_VLA.3.1C The vulnerability analysis documentation shall describe the analysis of the TOE deliverables performed to search for ways in which a user can violate the TSP.

AVA_VLA.3.2C The vulnerability analysis documentation shall describe the disposition of identified vulnerabilities.

AVA_VLA.3.3C The vulnerability analysis documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.3.4C The vulnerability analysis documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks.

AVA_VLA.3.5C The vulnerability analysis documentation shall show that the search for vulnerabilities is systematic.

AVA_VLA.3.1E **Refinement:** The NSA evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.3.2E **Refinement:** The NSA evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure the identified vulnerabilities have been addressed.

AVA_VLA.3.3E **Refinement:** The NSA evaluator shall perform an independent vulnerability analysis.

AVA_VLA.3.4E **Refinement:** The NSA evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of additional identified vulnerabilities in the intended environment.

AVA_VLA.3.5E **Refinement:** The NSA evaluator shall determine that the TOE is resistant to penetration attacks performed by an attacker possessing a moderate attack potential.

AVA_VLA.3.6E **Refinement:** The NSA evaluator shall perform an independent vulnerability analysis and conduct independent penetration testing.

7. Rationale

82 This section provides the rationale for the selection, creation, and use of security objectives and requirements as defined in sections 4 and 5, respectively.

7.1 Security Objectives derived from Threats

83 Each of the identified threats to security is addressed by one or more security objectives. Table 7.1 below provides the mapping from security objectives to threats, as well as a rationale that discusses how the threat is addressed. Definitions are provided (*in italics*) below each threat and security objective so the PP reader can reference these without having to go back to sections 3 and 4.

Table 7.1 – Mapping of Security Objectives to Threats

Threat	Objectives Addressing Threat	Rationale
T.ADMIN_ERROR *An administrator may incorrectly install or configure the TOE resulting in ineffective security mechanisms.*	O.ADMIN_GUIDANCE *The TOE will provide administrators with the necessary information for secure management of the TOE.* O.INSTALL_GUIDANCE *The TOE will be delivered with the appropriate installation guidance to establish and maintain TOE security.* O.MANAGE *The TOE will provide all the functions and facilities necessary to support the authorized administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.*	Improper or insufficient security policies and mechanisms might be implemented if the administrator is not properly trained. However, if the administrator is provided sufficient guidance for the installation [O.INSTALL_GUIDANCE], configuration, and management of the TOE [O.ADMIN_GUIDANCE], the threat that the administrator may incorrectly install, configure, or manage the TOE, in a way that undermines security, is reduced. O.MANAGE also contributes to mitigating this threat by providing the security mechanisms (e.g., tools for reviewing audit data) for administrators to perform TOE administration effectively, and to quickly alert the administrator of ineffective security policies on the TOE.
T.ADMIN_ROGUE *An authorized administrator's intentions may become malicious resulting in user or TSF data being compromised.*	O.ADMIN_ROLE *The TOE will provide administrator roles to isolate administrative actions.*	It is important to limit the functionality of administrative roles. If the intentions of an individual in an administrative role become malicious, O.ADMIN_ROLE mitigates this threat by isolating the administrative actions within that role and limiting the functions available to that individual. This objective presumes that separate individuals will be assigned separate distinct roles with no overlap of allowed operations between the roles. Separate roles include an authorized administrator and a cryptographic administrator.

Threat	Objectives Addressing Threat	Rationale
T.AUDIT_COMPROMISE *A malicious user or process may view audit records, cause audit records to be lost or modified, or prevent future records from being recorded, thus masking a user's actions.*	OE.PHYSICAL *Physical security will be provided for the TOE by the IT environment, commensurate with the value of the IT assets protected by the TOE.* O.AUDIT_GENERATION *The TOE will provide administrators with the necessary information for secure management of the TOE.* O.AUDIT_PROTECTION *The TOE will provide the capability to protect audit information.* O.REFERENCE_MONITOR *The TOE will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure.*	O.AUDIT_GENERATION provides the capability to detect and create records of security relevant events. Audit records identify the user responsible for the event and are an important form of evidence that can be used to track an attacker's actions. Tampering with or destruction of audit data by physical means is addressed by OE.PHYSICAL, which provides physical security controls to the TOE environment. O.AUDIT_PROTECTION provides the capability to specifically protect audit information from external interference, tampering, or unauthorized disclosure. O.REFERENCE_MONITOR protects the TOE and its resources (including audit data) by ensuring that the security policies implemented by the TOE to protect the audit information are always invoked.
T.CRYPTO _COMPROMISE *A malicious user or process may cause key, data or executable code associated with the cryptographic functionality to be inappropriately accessed (viewed, modified, or deleted), thus compromising the cryptographic mechanisms and the data protected by those mechanisms.*	OE.PHYSICAL *Physical security will be provided for the TOE by the IT environment, commensurate with the value of the IT assets protected by the TOE.* O.CRYPTOGRAPHIC_PROTECTION *The TOE will support separation of the cryptography from the rest of the TSF.* O.REFERENCE_MONITOR *The TOE will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure.*	The cryptography is afforded external protection from viewing, modification, or deletion by malicious users through physical security measures provided by the IT environment [OE.PHYSICAL]. Further, as part of the TOE's security functions (TSF), the cryptography is afforded internal protection from viewing, modification, or deletion by malicious processes and users through the domain isolation maintained by the TOE for its own execution [O.REFERENCE_MONITOR]. Within the TSF's domain an additional layer of protection (logical separation at a minimum) is applied to the cryptography [O.CRYPTOGRAPHIC_PROTECTION]. This additional layer helps to protect the cryptography against compromise from accidental interference (e.g. coding errors) and malicious untrusted subjects, however malicious parts of the kernel remain recognized threats unless stronger mechanisms are implemented to separate the cryptography.
T.EAVESDROP *A malicious user or process may observe or modify user or TSF data transmitted between physically separated parts of the TOE.*	O.ENCRYPTED_CHANNEL *Encryption will be used to provide confidentiality of TSF data in transit to remote parts of the TOE.*	The encryption of data before it is transmitted is the security measure that counters the ability to eavesdrop. O.ENCRYPTED_CHANNEL provides for the encryption of TSF data in transit between separate parts of the TOE, thereby defeating attempts by users or processes to intercept the transmitted data.

Threat	Objectives Addressing Threat	Rationale
T.POOR_DESIGN *Unintentional or intentional errors in requirements specification or design of the TOE may occur, leading to flaws that may be exploited by a malicious user or program.*	O.CHANGE_MANAGEMENT *The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled throughout the TOE's development.* O.SOUND_DESIGN *The TOE will be designed using sound design principles and techniques. The TOE design, design principles and design techniques will be adequately and accurately documented.* O.VULNERABILITY_ANALYSIS *The TOE will undergo appropriate vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with moderate attack potential to violate the TOE's security policies.*	Intentional or unintentional errors may occur in the requirement specification, design or development of the TOE. To address this threat, O.SOUND_DESIGN requires sound design principles and techniques that help prevent faults in the TOE's design by eliminating errors in the logic. In addition, O.CHANGE_MANAGEMENT addresses this threat by requiring all changes to the TOE and its development evidence be analyzed, tracked and controlled throughout the development cycle. To verify that there are no intentional or unintentional errors introduced in the design, O.VULNERABILITY_ANALYSIS demonstrates that the design of the TOE is resistant to attacks that exercise these design and development errors.
T.POOR _IMPLEMENTATION *Unintentional or intentional errors in implementation of the TOE design may occur, leading to flaws that may be exploited by a malicious user or program.*	O.CHANGE_MANAGEMENT *The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled throughout the TOE's development.* O.FUNCTIONAL_TESTING *The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.* O.PENETRATION_TEST *The TOE will undergo independent penetration testing to demonstrate that the design and implementation of the TOE prevents users from violating the TOE's security policies.* O.SOUND_IMPLEMENTATION *The implementation of the TOE will be an accurate instantiation of its design.* O.VULNERABILITY_ANALYSIS *The TOE will undergo appropriate vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with moderate attack potential to violate the TOE's security policies.*	Intentional or unintentional errors may occur when implementing the design of the TOE. To address this threat, O.SOUND_IMPLEMENTATION ensures that the implementation is an accurate representation of the design. To ensure that an accurate representation of the design is maintained, O.CHANGE_MANAGEMENT ensures that all changes to the TOE and its development evidence are analyzed, tracked and controlled throughout the development cycle. To ensure that errors have not been introduced, O.FUNCTIONAL_TESTING validates that the TSF satisfies the security functional requirements. To further demonstrate that vulnerabilities are not present, both O.PENETRATION_TESTING and O.VULNERABILITY_ANALYSIS ensure correct implementation of the TOE.

Threat	Objectives Addressing Threat	Rationale
T.MASQUERADE *A malicious user, process, or external IT entity may masquerade as an authorized entity to gain unauthorized access to data or TOE resources.*	O.USER_AUTHENTICATION *The TOE will verify the claimed identity of users.* O.USER_IDENTIFICATION *The TOE will uniquely identify users.*	To address this threat, O.USER_IDENTIFICATION identifies the user as a legitimate user and O.USER_AUTHENTICATION authenticates this user preventing unauthorized users, processes, or external IT entities from masquerading as an authorized entity.
T.POOR_TEST *Lack of or insufficient tests to demonstrate that all TOE security functions operate correctly may result in incorrect TOE behavior being undiscovered thereby causing potential security vulnerabilities.*	O.CORRECT_TSF_OPERATION *The TOE will provide a capability to test the TSF to ensure the correct operation of the TSF in its operational environment.* O.FUNCTIONAL_TESTING *The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.* O.PENETRATION_TEST *The TOE will undergo independent penetration testing to demonstrate that the design and implementation of the TOE prevents users from violating the TOE's security policies.*	Design analysis determines that a TOE's documented design satisfies its security functional requirements. In order to ensure the TOE's design is correctly realized in its implementation, the appropriate level of functional testing of the TOE's security mechanisms must be performed during the evaluation of the TOE. O.FUNCTIONAL_TESTING ensures that adequate functional testing is performed to demonstrate the TSF satisfies the security functional requirements and the TOE's security mechanisms operate as documented. While functional testing serves an important purpose, it does not ensure the TSFI cannot be used in unintended ways to circumvent the TOE's security policies. O.PENETRATION_TEST addresses this concern by requiring a vulnerability analysis be performed in conjunction with testing that goes beyond functional testing. This objective provides a measure of confidence that the TOE does not contain security flaws that may not be identified through functional testing. While these testing activities are a necessary activity for successful completion of an evaluation, this testing activity does not address the concern that the TOE continues to operate correctly and enforce its security policies once it has been fielded. Some level of testing must be available to authorized users to ensure the TOE's security mechanisms continue to operate correctly once the TOE is fielded. O.CORRECT_TSF_OPERATION ensures that once the TOE is installed at a customer's location, the capability exists that the integrity of the TSF (hardware and software) can be demonstrated, and thus provides end users the confidence that the TOE's security policies continue to be enforced.

Threat	Objectives Addressing Threat	Rationale
T.REPLAY *A user may gain inappropriate access to the TOE by replaying authentication information, or may cause the TOE to be inappropriately configured by replaying TSF data or security attributes.*	O.REPLAY_DETECTION *The TOE will provide a means to detect and reject the replay of authentication data, as well as, TSF data and security attributes.*	A common security threat is the interception and replay of security relevant information causing undesirable results. To prevent the negative effects of this threat, the TOE must provide mechanisms to ensure appropriate protection of security relevant data while it is in transit. Specifically, the TOE must detect and prevent the replay of an intercepted copy of protected authentication data as well as protected TSF data, such as security-relevant configuration parameters, that could cause the TOE to enter a state not intended by the TOE security administrator. The TOE objective O.REPLAY_DETECTION addresses this threat by ensuring that transmitted TSF data cannot be captured by a malicious user and resubmitted.
T.RESIDUAL_DATA *A user or process may gain unauthorized access to data through reallocation of TOE resources from one user or process to another.*	O.RESIDUAL_INFORMATION *The TOE will ensure that any data contained in a protected resource is not available when the resource is reallocated.*	The sharing of hardware resources such as primary and secondary storage components between users introduces the potential for information flow in violation of the TOE security policy when hardware resources are deallocated from one user and allocated to another. In order to prevent such unintended consequences, the TOE prevents the compromise of the TOE security policy through mechanisms that ensure that residual information cannot be accessed after the resource has been reallocated (O.RESIDUAL_INFORMATION). The intent here is to prevent the unauthorized flow of information that would violate the TOE security policy. The intent is not to require explicit scrubbing or overwriting of data prior to reuse of the storage resource. Therefore, the presence of “residual” data in a storage resource is acceptable as long as it cannot be accessed by subsequent users such that a violation of the TOE security policy results. Note, however, that the requirements for storage resources which contain critical cryptographic security parameters differ from the requirements for other types of data. Refer to the appropriate threat, objectives, and requirements rationale for a discussion of the requirements for residual data protection involving critical cryptographic security parameters.

Threat	Objectives Addressing Threat	Rationale
T.RESOURCE _EXHAUSTION *A malicious process or user may block others from system resources (i.e., system memory, persistent storage, and processing time) via a resource exhaustion denial of service attack.*	O.RESOURCE_SHARING *The TOE shall provide mechanisms that mitigate user attempts to exhaust TOE resources (e.g., system memory, persistent storage, and processing time).*	The sharing of resources (i.e., system memory, persistent storage, and processing time) between users introduces the potential for a malicious process or user to obstruct users from access to resources via a resource exhaustion denial-of-service attack. O.RESOURCE_SHARING mitigates this threat by requiring the TOE to provide controls to enforce maximum quotas for system memory, persistent storage, and processing time.
T.SPOOFING *A malicious user, process, or external IT entity may misrepresent itself as the TOE to obtain authentication data.*	O.TRUSTED_PATH *The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data.*	It is possible for an entity other than the TOE (a subject on the TOE, or another IT entity) to provide an environment that may lead a user to mistakenly believe they are interacting with the TOE, thereby fooling the user into divulging identification and authentication information. O.TRUSTED_PATH mitigates this threat by ensuring users have the capability to ensure they are communicating with the TOE when providing identification and authentication data to the TOE (e.g. initial login, unlocking a session, changing a password).
T.TSF_COMPROMISE *A malicious user or process may cause TSF data or executable code to be inappropriately accessed (viewed, modified, or deleted).*	OE.PHYSICAL *Physical security will be provided for the TOE by the IT environment, commensurate with the value of the IT assets protected by the TOE.* O.REFERENCE_MONITOR *The TOE will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure.*	The tampering with or destruction of TSF hardware, software, or configuration data via physical means is addressed by the physical security controls present in the TOE environment [OE.PHYSICAL]. O.REFERENCE_MONITOR addresses the threat of tampering with or destruction of TSF hardware, software, or configuration data by other (non-physical) means. It ensures that the TSF maintains a security domain for its own execution that protects it from interference and tampering by untrusted subjects and enforces the separation between the security domains of subjects within the TSC.
T.UNATTENDED _SESSION *A user may gain unauthorized access to an unattended session.*	O.PROTECT *The TOE will provide mechanisms to protect user data and resources.* O.TRAINED_USERS *The TOE will provide authorized users with the necessary guidance for secure use of the TOE, to include secure sharing of user data.*	When an authorized user leaves an active session unattended, an unauthorized user may gain access to the unattended session. O.PROTECT mitigates this threat by providing mechanisms to protect user data and resources from unauthorized access by ensuring that the TSF will lock an interactive session and make the visible contents unreadable after a specified time interval of session inactivity. In addition, the TSF also allows authorized users to lock their interactive session before leaving the session unattended [O.TRAINED_USERS].

Threat	Objectives Addressing Threat	Rationale
T.UNAUTHORIZED _ACCESS *A user may gain unauthorized access (view, modify, delete) to user data.*	OE.PHYSICAL *Physical security will be provided for the TOE by the IT environment, commensurate with the value of the IT assets protected by the TOE.* O.ACCESS *The TOE will ensure that users gain only authorized access to it and to resources that it controls.* O.ACCESS_HISTORY *The TOE will display information (to authorized users) related to previous attempts to establish a session.* O.PROTECT *The TOE will provide mechanisms to protect user data and resources.*	Unauthorized users may physically access TOE resources. To mitigate this threat, OE.PHYSICAL restricts the physical access only to authorized personnel. Within the computing environment, O.ACCESS restricts all access controls to authorized users based on their user identity. At the same time, O.PROTECT enforces access rules by providing mechanisms to prevent the user data from unauthorized disclosure and modification. O.ACCESS_HISTORY helps users confirm their previously established session or may help detected possible unsuccessful attempts to their account by an unauthorized user.
T.UNIDENTIFIED _ACTIONS *The administrator may fail to notice potential security violations, thus preventing the administrator from taking action against a possible security violation.*	O.AUDIT_REVIEW *The TOE will provide the capability to selectively view audit information and alert the administrator of identified potential security violations.* O.ADMIN_GUIDANCE *The TOE will provide administrators with the necessary information for secure management of the TOE.*	The threat of an administrator failing to know about audit events may occur. To mitigate this threat, O.AUDIT_REVIEW provides the capability to selectively view audit information, and alert the administrator of identified potential security violations. If alerted, the administrator needs to acknowledge the message and act according to the guidance [O.ADMIN_GUIDANCE].
T.UNKNOWN_STATE *When the TOE is initially started or restarted after a failure, the security state of the TOE may be unknown.*	O.RECOVERY *Procedures and/or mechanisms will be provided to assure that recovery is obtained without a protection compromise, such as from system failure or discontinuity.* O.SECURE_STATE *The TOE will be able to verify the integrity of the TSF code and cryptographic data.*	After a failure, the security condition of the TOE may be unknown. To mitigate this threat O.RECOVERY provides procedures and/or mechanisms to ensure that recovery without a protection compromise is obtained. O.SECURE_STATE provides the mechanisms to verify the correctness of the TSF code and data thus ensuring a secure state after a failure or upon startup.

7.2 Objectives derived from Security Policies

109 Each of the identified security policies is addressed by one or more security objectives. Table 7.2 below provides the mapping from security objectives to security policies, as well as a rationale that discusses how the policy is addressed. Definitions are provided (*in italics*) below each threat and security objective so the PP reader can reference these without having to go back to sections 3 and 4.

Table 7.2 – Mapping of Security Objectives to Security Policies

Security Policy	Objectives Addressing Policy	Rationale
P.ACCESS_BANNER *The TOE shall display an initial banner describing restrictions of use, legal agreements, or any other appropriate information to which users consent by accessing the TOE.*	O.DISPLAY_BANNER *The TOE will display an advisory warning regarding use of the TOE.*	O.DISPLAY_BANNER satisfies this policy by ensuring that the TOE displays a banner that provides authorized users with an advisory warning about the unauthorized use of the TOE.
P.ACCOUNTABILITY *The users of the TOE shall be held accountable for their actions within the TOE*	O.AUDIT_GENERATION *The TOE will provide administrators with the necessary information for secure management of the TOE.* O.AUDIT_REVIEW *The TOE will provide the capability to selectively view audit information and alert the administrator of identified potential security violations.* O.USER_IDENTIFICATION *The TOE will uniquely identify users.*	Enforcement of this policy requires that users be uniquely identified [O.USER_IDENTIFICATION] and that their security relevant actions be monitored and recorded [O.AUDIT_GENERATION]. The recorded audit information can be selectively reviewed in search of any potential security violations [O.AUDIT_REVIEW].
P.AUTHORIZATION *The TOE shall limit the extent of each user's abilities in accordance with the TSP.*	O.ACCESS *The TOE will ensure that users gain only authorized access to it and to resources that it controls.* O.PROTECT *The TOE will provide mechanisms to protect user data and resources.* O.USER_IDENTIFICATION *The TOE will uniquely identify users.*	O.ACCESS supports this policy by requiring the TOE to uniquely identify authorized users [O.USER_IDENTIFICATION] prior to allowing any TOE access or any TOE mediated access on behalf of those users. Within the TOE, O.PROTECT provides mechanisms to prevent user data from unauthorized disclosure and modification.

Security Policy	Objectives Addressing Policy	Rationale
P.AUTHORIZED_USERS *Only those users who have been authorized to access the information within the TOE may access the TOE.*	O.ACCESS *The TOE will ensure that users gain only authorized access to it and to resources that it controls.*	Within the set of all the users that may interact with the TOE, authorized users are those with access to the information within the TOE after being successfully identified and authenticated by the TOE. Access control policies are used to define the access permitted to the system and its resources. These policies are supported by the implementation of authorized user attributes that identify the user-allowed accesses to TOE information. O.ACCESS supports this policy by ensuring that users only gain authorized access to TOE information and its resources by checking user attributes before system use.
P.CRYPTOGRAPHY *The TOE shall use NIST FIPS validated cryptography as a baseline with additional NSA-approved methods for key management (i.e., generation, access, distribution, destruction, validation and packaging, handling, and storage of keys) and for cryptographic operations (i.e., encryption, decryption, signature, hashing, key exchange, and random number generation services).*	O.CRYPTOGRAPHIC_SERVICES *The TOE will make cryptographic services available to authorized users and/or user applications.*	By building upon NIST FIPS-validated, cryptography, the TOE not only provides, but also augments the cryptographic support offered solely by baseline NIST FIPS-validated cryptography. The TOE cryptography supports key management (i.e., generation, access, distribution, destruction, handling, and storage of keys) and cryptographic operations (i.e., encryption, decryption, signature, hashing, key exchange, and improved random number generation). O.CRYPTOGRAPHIC_SERVICES provides these cryptographic services to TOE authorized users and/or user applications.
P.I_AND_A *All users must be identified and authenticated prior to accessing any controlled resources with the exception of public objects.*	O.USER_AUTHENTICATION *The TOE will verify the claimed identity of users.* O.USER_IDENTIFICATION *The TOE will uniquely identify users.*	In support of the policy to identify and authenticate a user before access is granted to any controlled resources, O.USER_IDENTIFICATION and O.USER_AUTHENTICATION will uniquely identify and authenticate the claimed authorized users.

Security Policy	Objectives Addressing Policy	Rationale
P.INDEPENDENT_TESTING *The TOE must undergo independent testing as part of an independent vulnerability analysis.*	O.FUNCTIONAL_TESTING *The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.* O.PENETRATION_TEST *The TOE will undergo independent penetration testing to demonstrate that the design and implementation of the TOE prevents users from violating the TOE's security policies.* O.VULNERABILITY_ANALYSIS *The TOE will undergo appropriate vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with moderate attack potential to violate the TOE's security policies.*	This policy requires the TOE to undergo independent testing to verify its reliability and security. O.FUNCTIONAL_TESTING demonstrates the TSF satisfies the appropriate security functional requirements. O.PENETRATION_TESTING requires the TOE to undergo penetration testing and demonstrate that the design and implementation of the TOE do not allow users to violate the TOE's security policies. O.VULNERABILITY_ANALYSIS requires the TOE to undergo vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.
P.NEED_TO_KNOW *The TOE must limit the access to data in protected resources to those authorized users who have a need to know that data.*	O.ACCESS *The TOE will ensure that users gain only authorized access to it and to resources that it controls.* O.DISCRETIONARY_ACCESS *The TOE will control access to resources based upon the identity of users and groups of users.* O.DISCRETIONARY_USER_CONTROL *The TOE will allow authorized users to specify which resources may be accessed by which users and groups of users.* O.PROTECT *The TOE will provide mechanisms to protect user data and resources.*	The need-to-know policy is satisfied by the discretionary access control rules. O.DISCRETIONARY_ACCESS protects resources based on the identity of authorized users where the access to objects is directed by owners of the object [O.DISCRETIONARY_USER_CONTROL]. O.PROTECT enforces these policy rules by providing the mechanisms to protect the user data from disclosure and modifications and lastly, O.ACCESS ensures that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed.

Security Policy	Objectives Addressing Policy	Rationale
P.RATINGS_MAINTENANCE *A plan for procedures to maintain the TOE's rating must be in place.*	O.RATINGS_MAINTENANCE *Procedures to maintain the TOE's rating will be documented.*	O.RATINGS_MAINTENANCE satisfies this policy by ensuring that the TOE developer has procedures and mechanisms in place to maintain the evaluated rating that is ultimately awarded the TOE. The developer must provide a plan that identifies the certified version of the TOE and its life cycle process. Identifies any plans for new releases of the TOE to include a description of the changes included in the new release and a security impact analysis of implementing the new changes. Assign and identify the TOE's developer security analyst and ensure that they follow documented procedures. TOE components must be categorized by security relevance. The categorization scheme must be documented and followed for changes to the TOE.
P.REMOTE_ADMIN_ACCESS *Any remote administration shall be securely managed by the TOE.*	O.ENCRYPTED_CHANNEL *Encryption will be used to provide confidentiality of TSF data in transit to remote parts of the TOE* O.TSF_CRYPTOGRAPHIC_INTEGRITY *The TOE will provide cryptographic integrity mechanisms for TSF data while in transit to remote parts of the TOE.*	Remote administration is not a required functionality that the TOE must meet, but the PP authors recognize the operational need for remote administration in many distributed environments. For those TOEs that provide remote administration, it is very important that this functionality is managed securely. This policy requires the TOE to provide the capability to be remotely administered. To securely perform this policy, the system must protect all TSF data on this communication path during the remote administrative session. For secure remote administration capabilities, cryptographic mechanisms will be applied to maintain TSF data confidentiality and integrity. O.ENCRYPTED_CHANNEL and O.TSF_CRYPTOGRAPHIC_INTEGRITY provide the necessary protection of the TSF data on this communication path.
P.ROLES *The TOE shall provide multiple administrative roles for secure administration of the TOE. These roles shall be separate and distinct from each other.*	O.ADMIN_ROLE *The TOE will provide administrator roles to isolate administrative actions.*	To appropriately administer the system, O.ADMIN_ROLE requires the system to provide multiple administrator roles to isolate actions performed by these different roles. To completely satisfy this policy, separate roles must be assigned separate individuals.

Security Policy	Objectives Addressing Policy	Rationale
P.SYSTEM_INTEGRITY *The TOE shall provide the ability to periodically validate its correct operation and, with the help of administrators, it must be able to recover from any errors that are detected.*	O.CORRECT_TSF_OPERATION *The TOE will provide a capability to test the TSF to ensure the correct operation of the TSF in its operational environment.* O.RECOVERY *Procedures and/or mechanisms will be provided to assure that recovery is obtained without a protection compromise, such as from system failure or discontinuity.*	In order for an organization to place a measure of trust in the security features of a TOE, the TOE must include mechanisms that provide some measure of confidence in its correct functioning during its operational life-cycle. To provide such confidences, O.TRUSTED_SYSTEM_OPERATION provides self-tests that run during system start up, or at the request of the system administrator, and ensure that the TOE security mechanisms are operating properly. When a security failure occurs and the TOE self-tests determine that the TOE is not operating in accordance with its security policies, O.RECOVERY provides the mechanisms that will return the TOE to a known secure operating state such that the security policies are enforced on all future processing.
P.TRACE *The TOE shall provide the ability to review the actions of individual users.*	O.AUDIT_REVIEW *The TOE will provide the capability to selectively view audit information and alert the administrator of identified potential security violations.*	A common organizational security policy is to maintain records allowing for individuals to be held responsible for the actions that they take with respect to organizational assets. Information can be one of the most valuable assets that an organization possesses. To satisfy this policy, O.AUDIT_REVIEW provides suitable mechanisms to accurately and selectively review those records by authorized personnel to provide accountability at the individual user level to determine any potential security violation.
P.TRUSTED_RECOVERY *Procedures and/or mechanisms shall be provided to assure that, after a TOE failure or other discontinuity, recovery without a protection compromise is obtained.*	O.RECOVERY *Procedures and/or mechanisms will be provided to assure that recovery is obtained without a protection compromise, such as from system failure or discontinuity.*	After a failure or other discontinuity, the security condition of the TOE may be unknown. O.RECOVERY provides procedures and/or mechanisms to ensure that recovery to a known secure state is obtained without a protection compromise.
P.VULNERABILITY _ANALYSIS_AND_TEST *The TOE must undergo a vulnerability analysis and penetration testing by NSA to demonstrate that the TOE is resistant to an attacker possessing a moderate attack potential.*	O.VULNERABILITY_ANALYSIS *The TOE will undergo appropriate vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with moderate attack potential to violate the TOE's security policies.*	O.VULNERABILITY_ANALYSIS satisfies this policy by ensuring that an independent analysis is performed on the TOE and penetration testing based on that analysis is performed. Having an independent party perform the analysis helps ensure objectivity and eliminates preconceived notions of the TOE's design and implementation that may otherwise affect the thoroughness of the analysis. The level of analysis and testing requires that an attacker with a moderate attack potential cannot compromise the TOE's ability to enforce its security policies.

7.3 Objectives derived from Assumptions

130 Each of the identified security assumptions is addressed by one or more security objectives. Table 7.3 below provides the mapping from security objectives to security policies, as well as a rationale that discusses how the policy is addressed. Definitions are provided (*in italics*) below each threat and security objective so the PP reader can reference these without having to go back to sections 3 and 4.

Table 7.3 – Mapping of Security Objectives to Assumptions

Assumption	Objectives Addressing Assumption	Rationale
A.PHYSICAL *It is assumed that the IT environment provides the TOE with appropriate physical security, commensurate with the value of the IT assets protected by the TOE .*	OE.PHYSICAL *Physical security will be provided for the TOE by the IT environment, commensurate with the value of the IT assets protected by the TOE.*	Physical security must be provided for the TOE by the IT environment to ensure the TOE is capable of addressing the threats to TOE assets [OE.PHYSICAL].

7.4 Requirements Rationale

132 Each of the security objectives identified in sections 7.1 and 7.2 are addressed by one or more security requirements. Table 7.4 below provides the mapping from security requirements to security objectives, as well as a rationale that discusses how the security objective is met. Definitions are provided (*in italics*) below each security objective so the PP reader can reference these without having to go back to section 4.

Table 7.4 – Mapping of Security Requirements to Objectives

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
O.ACCESS *The TOE will ensure that users gain only authorized access to it and to resources that it controls.*	FDP_ACC.2 FDP_ACF.1 FIA_AFL.1 FIA_ATD.1 FMT_REV.1(1) FMT_REV.1(2) FPT_RVM.1 FPT_TRC_EXP.1 FTA_LSA.1 FTA_MCS.1 FTA_SSL.1 FTA_SSL.2 FTA_TSE.1	The TOE must protect itself and the resources it controls from unauthorized access. FDP_ACC.2 enforces the Discretionary Access Control (DAC) policy on all subjects and all named objects and all operations among them. The DAC policy specifies the access rules between all subjects and all named objects controlled by the TOE. While authorized users are trusted to some extent, this requirement ensures only authorized access is allowed to named objects. FDP_ACF.1 specifies the DAC policy rules that will be enforced by the TSF and determines if an operation among subjects and named objects is allowed. Furthermore, it specifies the rules to explicitly authorize or deny access to a named object based upon security attributes. FIA_AFL.1 provides a detection mechanism for unsuccessful authentication attempts. The requirement enables an authorized administrator configurable threshold that prevents unauthorized users from gaining access to authorized user's account by guessing authentication data. This mechanism prevents access by either disabling the targeted account. Thus, limiting an unauthorized user's ability to gain unauthorized access to the TOE. FIA_ATD.1 defines the attributes of users, including a userid that is used by the TOE to determine a user's identity and enforce what type of access the user has to the TOE (e.g., the TOE associates a userid with any role(s) they may assume). FMT_REV.1(1) ensures that the authorized administrator has the ability to revoke security attributes to a specific user. This revocation is immediate and helps authorized administrators control the ability of authorized users to log in or perform privileged operations. FMT_REV.1(2) ensures that the authorized administrator and owners of named objects have the ability to revoke security attributes to a specific user. This revocation occurs when an access check is made and helps authorized administrators and owners control the ability of users accessing named objects. FPT_RVM.1 ensures that the TSF makes policy decisions on all access attempts to the TOE resources. Without this non-bypassability

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		requirement, the TSF could not be relied upon to completely enforce the security policies. While untrusted users are only intended to access public objects, this requirement ensures they cannot access other objects provided by the TOE. This requirement also ensures that an administrator acting in a role only has access to the functions designated for that role. FPT_TRC_EXP.1 ensures that the TSF data is consistent between parts of the TOE by providing a mechanism to bring inconsistent TSF data into a consistent state in a timely manner. Such data may become inconsistent if an internal channel between parts of the TOE becomes inoperative or in the case of a distributed TOE, this can occur when parts become disabled, network connections are broken, and so on. The ability to ensure that the TSF data is consistent, between parts of the TOE, affords the TOE the ability to maintain the security policies current throughout all parts of the TOE and limits the opportunity of an outdated security policy to be enforced on parts of the TOE that may be permitting unauthorized access to the TOE and its resources. FTA_LSA.1 ensures that the scope of roles, user sensitivity and integrity levels and user privileges are restricted based on location, time, and day. With the distributed nature of the systems today, limitations need to be defined to control access and privileged functions. For example, a security policy may be set to prohibit access to the highest level of sensitivity or to the most privileged functions from certain locations. FTA_MCS.1 is used to control the ability of authorized users to establish more TOE sessions than the maximum number of concurrent interactive sessions allowed. The ability of the administrator to determine how many sessions are allowed affords the TOE the ability to limit the exposure of the TOE to an attacker attempting to establish a session after the maximum number of allowed sessions is met. FTA_SSL.1 is used to prevent unauthorized access to the TOE and its resources when an interactive session is left unattended. This requirement ensures that the interactive session will lock by making the visible contents unreadable after a specified time interval of session inactivity. The authorized user needs to re-authenticate to unlock his session. FTA_SSL.2 is used to ensure that unauthorized access to the TOE and its resources when an interactive session is left unattended. It enables the authorized user to lock his interactive session before leaving the session unattended. This eliminates any chance for any user to acquire unauthorized access to an unattended session because there is no time interval of inactivity before the session is locked. The authorized user needs to re-authenticate to unlock his session. FTA_TSE.1 is used to control the ability of an authorized user to establish a TOE session. The ability of a the administrator to determine which users are able to establish a session at a specific range of time, and from a specific location affords the TOE the ability to limit the exposure of the TOE to an attacker attempting to establish

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		a session. For example, if the authorized user John Doe is only allowed to establish a session from 8 to 5, Monday through Friday, anyone attempting to establish a session as John Doe other than during those hours would not succeed, regardless of possession of John Doe's authentication data.
O.ACCESS_HISTORY *The TOE will display information (to authorized users) related to previous attempts to establish a session.*	FTA_TAH.1	FTA_TAH.1 is used to provide information about previous interactive sessions (i.e., date, time, and location). This information is displayed to the authorized user upon each successful interactive session establishment. This requirement gives the authorized users the ability to verify their last successful interactive session and thus, is a means for determining if the previous successful interactive session establishment was authorized or not.
O.ADMIN_ROLE *The TOE will provide administrator roles to isolate administrative actions.*	FMT_SMR.2 FMT_SMR.3	The TOE must maintain roles to isolate administrative actions. FMT_SMR.2 ensures that a minimum of two administrative roles are maintained (i.e., authorized administrator and cryptographic administrator) and that no overlapping of operations exists between roles. FMT_SMR.3 requires authorized users to make explicit requests to assume any administrative role. Every time an administrative role is assumed, this event will be audited.
O.ADMIN_GUIDANCE *The TOE will provide administrators with the necessary information for secure management.*	ADO_IGS.1 AGD_ADM.1	ADO_IGS.1 provides the procedures necessary for the secure installation, generation, and start-up of the TOE. AGD_ADM.1 provides administrative guidance to configure and administer the TOE securely for the IT environment it is intended to operate. The guidance also provides information about the corrective measures necessary when a failure occurs (i.e., how to bring the TOE back into a secure state).
O.AUDIT_GENERATION *The TOE will provide the capability to detect and create records of security relevant events associated with users.*	FAU_GEN.1 FAU_GEN.2 FAU_SEL.1 FIA_AFL.1 FIA_USB.1 FPT_STM.1	FAU_GEN.1 defines the set of events that the TOE must be capable of recording. This requirement ensures that the authorized administrator has the ability to audit any security relevant event that takes place in the TOE. This requirement also defines the information that must be contained in the audit record for each auditable event. There is a minimum of information that must be present in every audit record and this requirement defines that, as well as the additional information that must be recorded for each auditable event. This requirement also places a requirement on the level of detail that is recorded on any additional security functional requirements an ST author adds to this PP. FAU_GEN.2 ensures that the audit records associate a user identity with the auditable event. The association is accomplished using the userid of the authorized user. FAU_SEL.1 allows the authorized administrator to configure which auditable events will be recorded in the audit trail. This provides the administrator with the flexibility in recording only those events that are deemed necessary by site policy, thus reducing the amount of

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		resources consumed by the audit mechanism. FIA_USB.1 plays a role is satisfying this objective by requiring a binding of security attributes associated with users that are authenticated with the subjects that represent them in the TOE. This only applies to authenticated users, since the identity of unauthenticated users cannot be confirmed. Therefore, the audit trail may not always have the proper identity of the user that causes an audit record to be generated (e.g., an attacker/user providing another user's user identifier). FPT_STM.1 ensures that the time stamps used to create the audit records are reliable. The time and date included in the time stamp is crucial when generating the audit information to ensure accountability.
O.AUDIT_PROTECTION *The TOE will provide the capability to protect audit information.*	FAU_SAR.2 FAU_STG.1	The audit trail must be protected so that only authorized users and authorized administrators may access it or delete it. FAU_SAR.2 ensures that only authorized users have read access to audit information and FAU_STG.1 ensures that audit information is not modified and protects it from unauthorized deletions.
O.AUDIT_REVIEW *The TOE will provide the capability to selectively view audit information and alert the administrator of identified potential security violations.*	FAU_ARP.1 FAU_SAA.1 FAU_SAR.1 FAU_SAR.3 FAU_STG.4	FAU_SAA.1 defines the events that indicate a potential security violation and will generate an alarm. The triggers for these events are generally configurable by an authorized administrator. The events include at minimum authentication failures, Discretionary Access Control policy violation attempts, failures of the cryptographic self-tests and failures of the TSF self-tests. FAU_SAR.1 provides the ability for an authorized administrator to efficiently review audit records. This requirement also mandates the audit information be presented in a manner that is suitable for the administrators to interpret the audit trail. FAU_SAR.3 complements FAU_SAR.1 by providing the administrators the flexibility to specify criteria that can be used to search or sort the audit records residing in the audit trail. FAU_SAR.3 requires the administrators be able to establish the audit review criteria based on a user and identifier, date and time, so that the actions of a user can be readily identified and analyzed. Allowing the administrators to perform searches or sort the audit records based on dates, times, type of events, and success and failure of these events, provides the capability to extract the user activity to what is pertinent at that time in order facilitate the administrator's review. It is important to note that the intent of sorting in this requirement is to allow the administrators the capability to organize or group the records associated with a given criteria. FAU_ARP.1 and FAU_STG.4 allow the authorized administrator to be alerted upon the detection of a potential security violation and when the audit trial becomes full. The latter prevents the execution of an audit trail exhaustion attack.
O.CHANGE_MANAGEMENT	ACM_AUT.1	ACM_CAP.4 contributes to this objective by requiring the developer

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled throughout the TOE's development life-cycle.	ACM_CAP.4 ACM_SCP.2 ALC_DVS.1 ALC_FLR.2 ALC_LCD.1 ALC_TAT.1	to have a configuration management plan that describes how changes to the TOE and its evaluation deliverables are managed. The developer is also required to employ a configuration management system that operates in accordance with the CM plan and provides the capability to control who on the development staff can make changes to the TOE and its developed evidence. This requirement also ensures that authorized changes to the TOE have been analyzed and the developer's acceptance plan describes how this analysis is performed and how decisions to incorporate the changes to the TOE are made. ACM_AUT.1 complements ACM_CAP.4, by requiring that the CM system use an automated means to control changes made to the TOE. If automated tools are used by the developer to analyze, or track changes made to the TOE, those automated tools must be described. This aids in understanding how the CM system enforces the control over changes made to the TOE. ACM_SCP.2 is necessary to define what items must be under the control of the CM system. This requirement ensures that the TOE implementation representation, design documentation, test documentation (including the executable test suite), user and administrator guidance, CM documentation and security flaws are tracked by the CM system. ALC_DVS.1 requires the developer describe the security measures they employ to ensure the integrity and confidentiality of the TOE are maintained. The physical, procedural, and personnel security measures the developer uses provides an added level of control over who and how changes are made to the TOE and its associated evidence. ALC_FLR.2 plays a role in satisfying the "analyzed" portion of this objective by requiring the developer to have procedures that address flaws that have been discovered in the product, either through developer actions (e.g., developer testing) or those discovered by others. The flaw remediation process used by the developer corrects any discovered flaws and performs an analysis to ensure new flaws are not created while fixing the discovered flaws. ALC_LCD.1 requires the developer to document the life-cycle model used in the development and maintenance of the TOE. This life-cycle model describes the procedural aspects regarding the development of the TOE, such as design methods, code or documentation reviews, how changes to the TOE are reviewed and accepted or rejected. ALC_TAT.1 ensures that all the tools used during the development and maintenance of the TOE are well defined including the selected implementation-dependent options of the development tools.

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
O.CORRECT_TSF _OPERATION *The TOE will provide a capability to test the TSF to ensure the correct operation of the TSF in its operational environment.*	FMT_MSA.2 FPT_TST.1(1) FPT_TST.1(2) FPT_TST_EXP.1	This objective requires the FPT_TST requirement to be met:. This functional requirement provides the end user with the capability to ensure the TOE's security mechanisms continue to operate correctly in the field. FPT_TST_EXP.1 is necessary to ensure the correctness of the TSF software. If TSF software is corrupted, it is possible that the TSF would no longer be able to enforce the security policies. The FPT_TST.1(1) and FPT_TST.1(2) functional requirements address the critical nature and specific handling of the critical cryptographic security functions related to TSF data. Since the critical cryptographic security functions have specific requirements associated with them it is important to ensure that any fielded testing on the integrity of these data maintains the same level of security as specified in the FCS functional requirements. Additionally, O.CORRECT_TSF_OPERATION requires FMT_MSA.2. This requirement ensures that only valid values are accepted for security attributes. The values that are accepted as valid for a specific security attribute must fall within the appropriate range for that attribute (e.g., the password length attribute must be a non-negative integer).
O.CRYPTOGRAPHIC _PROTECTION *The TOE will support separation of the cryptography from the rest of the TSF.*	FPT_SEP.2	As part of the TOE's security functions (TSF), the cryptography is afforded separation and internal protection from viewing, modifications, or deletions by malicious processes and users through the domain isolation maintained by the TOE for its own execution [FPT_SEP.2]. At a minimum, the TSF provides logical separation for the part of the TSF implementing the cryptographic algorithms and persistent keys. This helps to protect the cryptography from interference and tampering by the remainder of the TSF and by subjects untrusted with respect to cryptography. This provides minimal separation of the cryptography within the kernel since it only protects the cryptography against accidental interference (e.g. coding errors) and malicious untrusted subjects. It does not protect the cryptography from any malicious part of the kernel. {Note: Stronger, preferred implementations such as, off board hardware or a third processor hardware state, would protect the cryptography from all other parts of the TSF.}

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
O.CRYPTOGRAPHIC_SERVICES *The TOE will make cryptographic services available to authorized users and/or user applications.*	FCS_CKM.1(1) FCS_CKM.1(2) FCS_CKM.2 FCS_CKM.4 FCS_CKM_EXP.1 FCS_CKM_EXP.2 FCS_BCM_EXP.1 FCS_COA_EXP.1 FCS_COP.1(1) FCS_COP.1(2) FCS_COP.1(3) FCS_COP.1(4) FCS_COP_EXP.1	Baseline cryptographic services are provided in the TOE by FIPS PUB 140-2 compliant modules implemented in hardware, in software, or in hardware/software combinations [FCS_BCM_EXP.1]. The cryptographic services offered by this baseline capability are augmented and customized in the TOE to support medium robustness environments. These TOE services are based primarily upon functional security requirements in the areas of key management and cryptographic operations. In the area of key management there are functional requirements that address the generation of symmetric keys [FCS_CKM.1 (1)], and the generation of asymmetric keys [FCS_CKM.1 (2)]; methods of manual and automated cryptographic key distribution [FCS_CKM.2]; cryptographic key destruction [FCS_CKM.4]; techniques for cryptographic key validation and packaging [FCS_CKM_EXP.1]; and cryptographic key handling and storage [FCS_CKM_EXP.2]. Specific functional requirements in the area of cryptographic operations address data encryption and decryption [FCS_COP.1 (1)]; cryptographic signatures [FCS_COP.1 (2)]; cryptographic hashing [FCS_COP.1 (3)]; cryptographic key agreement [FCS_COP.1 (4)]; and improved random number generation [FCS_COP_EXP.1]. These TOE requirements support cryptographic services that can be called upon by the TOE itself, or by TOE authorized users and/or user applications [FCS_COA_EXP.1].

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
O.DISCRETIONARY_ACCESS *The TOE will control access to resources based upon the identity of users and groups of users.*	FDP_ACC.2 FDP_ACF.1 FIA_USB.1 FMT_MSA.3 FPT_RVM.1 ADV_SPM.1	Access to TOE resources is determined by the Discretionary Access Control policy. FDP_ACC.2 ensures that the Discretionary Access Control policy is enforced on all subjects and all named objects and all operations between them. FDP_ACF.1 defines the Discretionary Access Control rules to determine if any operation between subjects and named objects is allowed. These rules are based on the identity of the users and their group memberships. FIA_USB.1 defines the associations between user security attributes and subjects acting on behalf of that user by which policy decisions are based upon. FMT_MSA.3 ensures that the TOE provides protection by default for all named objects at creation time. This may allow authorized users to explicitly specify the desired access controls upon the object at its creation, provided that there is no window of vulnerability through which unauthorized access may be gained to newly-created objects. FPT_RVM.1 ensures that the Discretionary Access Control policy is not bypassed. The discretionary aspect of the policy is that users who control access to objects can set that access to be restrictive or permissive to other users at their discretion. The policy is to be always enforced, never optional. ADV_SPM.1 requires the developer to provide an informal model of the Discretionary Access Control policy. Modeling the policy helps understand and reduce the unintended side-effects that occur during the TOE's operation that might adversely affect the TOE's ability to enforce its security policies.
O.DISCRETIONARY_USER_CONTROL *The TOE will allow authorized users to specify which resources may be accessed by which users and groups of users.*	FMT_MSA.1 FMT_REV.1(2)	To allow authorized users to specify which resources may be accessed, the TOE must provide the ability for object security attributes to be changed and revoked. FMT_MSA.1 restricts the ability to change the value of object security attributes to authorized administrators and owners of objects. FMT_REV.1(2) restricts the ability to revoke security attributes of named objects to authorized administrators and owners of these objects.
O.DISPLAY_BANNERS *The TOE will display an advisory warning regarding use of the TOE.*	FTA_TAB.1	Before identification and authentication and the establishment of a user session, the TOE allows limited access by any potential users of the system in order to convey warnings and agreements for system use. Through this limited access before establishing a user session, the TSF displays an authorized, administrator-specified advisory notice and consent warning message regarding unauthorized use of the TOE [FTA_TAB.1]. In typical applications a user who continues session establishment procedures (including their successful identification and authentication) after display of the notice and warning banner effectively acknowledges the banner content and consents to the stated conditions. This banner of information can be critical in supporting legal actions related to the use of the TOE.

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
O.ENCRYPTED_CHANNEL *Encryption will be used to provide confidentiality of TSF data in transit to remote parts of the TOE.*	FPT_ITT.1	The TOE protects basic internal transfers of TSF data. FPT_ITT.1 requires the TSF to use encryption to protect TSF data from disclosure when it is transmitted between separate parts of the TOE across an internal channel. Encryption is the security measure that provides data confidentiality (i.e. protects against disclosure) by translating the data into an unintelligible form before transmission.
O.FUNCTIONAL_TESTING *The TOE will undergo appropriate security functional testing, that demonstrates the TSF satisfies the security functional requirements.*	ATE_COV_EXP.2 ATE_DPT.2 ATE_FUN.1 ATE_IND.2	In order to satisfy O.FUNTIONAL_TESTING, the ATE class of requirements is necessary. Requirements fall into two categories; those that are levied on the developer to create and document the security test suite and those that are levied on the evaluation team to independently verify the testing results. The first category comprises ATE_FUN.1, ATE_COV_EXP.2, and ATE_DPT.2. The component ATE_FUN.1 requires the developer to provide the necessary test documentation to allow for an independent analysis of the developer's security functional test coverage. In addition, the developer must provide the test suite executables and source code, which are used for independently verifying the test suite results and in support of the test coverage analysis activities. ATE_COV_EXP.2 requires the developer to provide a test coverage analysis that demonstrates the TSFI are completely addressed by the developer's test suite. While exhaustive testing of the TSFI is not required, this component ensures that the security functionality of each TSFI is addressed. This component also requires an independent confirmation of the completeness of the test suite, which aids in ensuring that correct security relevant functionality of a TSFI is demonstrated through the testing effort. ATE_DPT.2 requires the developer to provide a test coverage analysis that demonstrates depth of coverage of the test suite. This component complements ATE_COV_EXP.2 by ensuring that the developer takes into account the high-level and low-level design when developing their test suite. Since exhaustive testing of the TSFI is not required, ATE_DPT.2 ensures that subtleties in TSF behavior that are not readily apparent in the functional specification are addressed in the test suite. The second category comprises ATE_IND.2 which requires an independent confirmation of the developer's test results, by mandating a subset of the test suite be run by an independent party. This component also requires an independent party to attempt to craft functional tests that address functional behavior that is not demonstrated in the developer's test suite. Upon successful adherence to these requirements, the TOE's conformance to the specified security functional requirements will have been demonstrated.
O.INSTALL_GUIDANCE *The TOE will be delivered with the appropriate installation guidance to establish and maintain TOE security.*	ADO_DEL.2 ADO_IGS.1	TOE delivery addresses the procedures that maintain security during any transfers of the TOE to the user. This includes transfers of the whole TOE or parts of the TOE upon initial delivery or upon any TOE upgrades/modifications. These procedures include measures to ensure that the security protection offered by the TOE is not compromised during the transfer. Specifically, ADO_DEL.2 [Detection of Modification] requires documented delivery procedures

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		for the TOE and TOE parts that describe how the procedures and technical measures provide for: (1.) the detection of modifications, (2.) detection of any discrepancy between the developer's master version and the delivered version, and (3.) detection of any attempts to masquerade as the developer. Additionally, ADO-DEL.2 requires: (1.) an evaluator to confirm (analyze) that the procedures meet all requirements for content and presentation of evidence, and (2.) the developer to follow the delivery procedures. Once secure delivery from the developer to the user has occurred, appropriate installation guidance should be used for the secure installation, generation and start-up of the TOE at the user's site. This phase securely transitions the TOE from the developer's configuration control to the user's operational environment. ADO_IGS.1 requires the developer to describe and document the procedures needed for secure TOE installation, generation, and start-up. ADO_IGS.1 also requires an evaluator to confirm that the procedures meet all the requirements for content and presentation of evidence, and that the procedures result in a secure configuration for the TOE.
O.MANAGE *The TOE will provide all the functions and facilities necessary to support the authorized administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.*	FMT_MOF.1(1) FMT_MOF.1(2) FMT_MSA.1 FMT_MSA.3 FMT_MTD.1(1) FMT_MTD.1(2) FMT_MTD.1(3) FMT_MTD.1(4) FMT_MTD.1(5) FMT_MTD.1(7) FMT_REV.1(1) FMT_REV.1(2) FMT_SAE.1 FMT_SMF.1	In a variety of ways the TOE supports authorized administrators in the management of security functions, security attributes and data while also restricting unauthorized use. For example, the TOE provides for and restricts the following actions to authorized administrators only (except where specifically noted): • Disable and enable the audit functions, and specify which events are audited [FMT_MOF.1 (1)]. • Create, initialize, change default, modify, delete, clear, append, query, etc. the values of security attributes associated with user authentication data [FMT_MOF.1 (2)]. • Change the value of object security attributes. (Object owner is also allowed to perform this action.) [FMT_MSA.1]. • Provide restrictive default values for security attributes, and specify alternative initial values to override the default values when an object or information is created. [FMT_MSA.3]. • Create, initialize, change default, modify, delete, clear, append, query, etc. the security-relevant TSF data (except audit records, user security attributes, authentication data, and critical security parameters) [FMT_MTD.1 (1)]. • Query, delete, and clear audit records [FMT_MTD.1 (2)]. • Initialize user security attributes. [FMT_MTD.1 (3)]. • Modify user security attributes, other than authentication data. [FMT_MTD.1 (4)]. • Modify authentication data. (Also allows users authorized to modify their own authentication data to do so.) [FMT_MTD.1 (5)].

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		• Revoke security attributes associated with the users within the TSC. [FMT_REV.1 (1)]. • Revoke security attributes of named objects within the TSC. (Object owner is also allowed to perform this action.) [FMT_REV.1 (2)]. • Specify an expiration time for authorized user authentication data. [FMT_SAE.1]. In addition, the TOE restricts the management of the critical cryptographic security parameters to cryptographic administrators [FMT_MTD.1 (7)]. FMT_SMF.1 provides a list of the management functions specified in this PP and is required as a dependency for the management functions.
O.PENETRATION_TEST *The TOE will undergo independent penetration testing to demonstrate that the design and implementation of the TOE prevents users from violating the TOE's security policies.*	AVA_VLA.3	O.PENETRATION_TESTING requires that the TOE have independent penetration testing performed on its design and implementation demonstrating that users cannot violate the TOE's security policies. AVA_VLA.3 requires the evaluator to conduct penetration testing and perform an independent vulnerability analysis to determine the exploitability of identified vulnerabilities and determine the resistance of the TOE to penetration attacks.
O.PROTECT *The TOE will provide mechanisms to protect user data and resources.*	FDP_ACC.2 FDP_ACF.1 FDP_RIP.2 FIA_SOS.1 FIA_UAU.7 FMT_MTD.1.1(6) FMT_REV.1(2) FPT_RVM.1 FPT_SEP.2	O.PROTECT requires mechanisms be provided by the TOE to protect user data and resources. FIA_SOS.1 prescribes the metrics that must be satisfied for user authentication. If a user can't authenticate, he or she will not have the ability to access user data and resources. FIA_SOS.1 requires that the authentication mechanism provide the ability for authorized users to have a "secret" in a manner that cannot be guessed at random in less than one in 5 x 1015. FIA_UAU.7 ensures that no feedback that affects the ability of users to circumvent the authentication mechanism is presented during the authentication process. The TOE is allowed to provide information that would allow the user to use the authentication mechanism in a correct manner (e.g., press CTRL-ALT-DELTE, slide card quickly, center your finger and press firmly, speak louder and slowly), but not provide information that may allow alteration to their presentation that would thwart the mechanism. FMT_MTD.1(6) ensures that the authentication data is protected. No entity is allowed to read authentication data and the TSF must prevent any attempt to read it. FPT_RVM.1 requires the TSF enforce a policy before each user action to protect resource in question. FPT_SEP.2 provides separation of data and resources so that untrusted subjects cannot access or manipulate other user data and resources in violation of the TOE policy. To protect user data and resources, FDP_ACC.2, FDP_ACF.1, and FMT_REV.1(2) require a Discretionary Access

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		policy and rules that ensures the correct access to named objects by subjects acting on behalf of users. To ensure that user data is not disclosed before a resource is reused, FDP_RIP.2 ensures that the user data contained within the object is not available to another user thus protecting the user data.
O.RATINGS _MAINTENANCE *Procedures to maintain the TOE's rating will be documented.*	AMA_AMP_EXP. 1	The AMA family of requirements is incorporated into this PP to ensure the TOE developer has procedures and mechanisms in place to maintain the evaluated rating that is ultimately awarded the TOE. These requirements are somewhat related to the ACM family of requirements in that changes to the TOE and its evidence must be managed, but the AMA requirements ensure the appropriate level of analysis is performed on any changes made to the TOE to ensure the changes do not affect the TOE's ability to enforce its security policies. AMA_AMP_EXP.1 requires the developer to develop an assurance maintenance (AM) plan that describes how the assurance gained from an evaluation will be maintained, and that any changes to the TOE will be analyzed to determine the security impact, if any, of the changes that are made. This requirement mandates the developer assign personnel to fulfill the role of a security analyst that is responsible for ensuring the changes made to the TOE will not adversely impact the TOE and that it will continue to maintain its evaluation rating.
O.RECOVERY *Procedures and/or mechanisms will be provided to assure that recovery is obtained without a protection compromise, such as from system failure or discontinuity.*	FCS_CKM_EXP.2 FPT_RCV.1 FPT_TRC_EXP.1 ADO_IGS.1	FPT_RCV.1 ensures that the system enters a maintenance mode allowing the system to be returned to a secure state after a failure or service discontinuity. In a secure state, all security policies are enforced; in addition, the critical areas of the cryptography are zeroized, are ready to be reloaded, and are inaccessible to processes. If the system needs to be recovered by re-installation, ADO_IGS.1 provides the documentation necessary to install, generate and start-up the TOE. To ensure that recovery is obtained without a protection compromise, it is critical that the information that all policy decisions are based on is correct and consistent. FPT_TRC_EXP.1 provides a mechanism to bring the TOE into a consistent state. TSF data may become inconsistent if an internal channel between parts of the TOE becomes inoperative or in the case of a distributed TOE, this can occur when parts become disabled, network connections are broken, and so on. The ability to ensure that the TSF data is consistent, between parts of the TOE, provides the TOE the ability to maintain the security policies current throughout all parts of the TOE and limits the opportunity of an outdated security policy to be enforced on parts of the TOE that may be permitting unauthorized access to the TOE and its resources. This requirement provides the mechanisms to ensure that upon reconnection, the TSF portions will become in sync over a reasonable time period.

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
O.REPLAY_DETECTION *The TOE will provide a means to detect and reject the replay of authentication data, as well as, TSF data and security attributes.*	FPT_ITT.3 FPT_RPL.1	O.REPLAY_DETECTION is satisfied by the requirements FPT_RPL.1 and FPT_ITT.3. These requirements ensure the TOE detects attempted replay of TSF data. These requirements ensure the TOE will audit the detection of replay and reject the data, which affords the administrators the opportunity to be aware of users attempting to replay critical data and affect the TOE's ability to enforce security policies.
O.RESIDUAL _INFORMATION *The TOE will ensure that any data contained in a protected resource is not available when the resource is reallocated.*	FCS_CKM.4 FCS_CKM_EXP.2 FDP_RIP.2	This objective is addressed by specifying requirements on two different types of resources in the TOE: cryptographic objects and all other recourses. For cryptographic objects, FCS_CKM_EXP.2 and FCS_CKM.4.1 ensure that cryptographic keys and critical cryptographic security parameters are protected. FCS_CKM_EXP.2 places requirements on how cryptographic keys are managed within the TOE. This requirement places restrictions when a cryptographic key is moved from one location to another (e.g., calculated in some scratch memory and moved to a permanent location) that the memory area is immediately cleared as opposed to waiting until the memory is reallocated to another subject. FCS_CKM.4 applies to the destruction of cryptographic keys used by the TSF. This requirement specifies how and when cryptographic keys must be destroyed. The proper destruction of these keys is critical in ensuring the content of these keys cannot possibly be disclosed when a resource is reallocated to a user. For all other resources, FDP_RIP.2 ensures that contents of resources are unavailable to subjects other than those explicitly granted access to the data.
O.RESOURCE_SHARING *The TOE shall provide mechanisms that mitigate user attempts to exhaust TOE resources (i.e., system memory, persistent storage, and processing time).*	FRU_RSA.1(1) FRU_RSA.1(2) FRU_RSA.1(3) FTA_MCS.1	This objective requires mechanisms to prevent authorized users (or software unknowingly acting on their behalf) from exhausting important resources controlled by the TOE in a manner that adversely impacts other users or programs. TOE is required to enforce a limit on the amount of resource a given authorized user may successfully be granted. The resources that are controlled are: CPU time, disk space, system memory, and user accounts. FRU_RSA.1 (iterations 1, 2 and 3) is intended to enforce the notion that a single authorized user may only be allocated a "preset maximum" amount of resource. The iterations cover the major resources that are required to offer confidence that entities executing on the TOE are not "starved for resource" and will be allowed to initiate and complete execution. 21 FTA_MCS.1 identifies user accounts as a system resource that could be exhausted (through multiple concurrent "logons" of a single individual). The requirement mandates that the administrator be able

[21] Note the requirement mandates that resource quotas be based on authorized users as opposed to "processes". This means that TOE mechanisms must enforce the policy across potentially multiple processes allocated to a single authorized user.

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		to limit the number of concurrent logon sessions by a single user. This ensures that a single individual could not mount a denial-of-service attack using multiple sessions as launching points. Resources (e.g., memory contained on the network card) that are not covered by the above are subject to denial of service attacks. Denial-of-service attacks of these resources should be addressed via other mechanisms such as redundant hardware.
O.REFERENCE_MONITOR *The TOE will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure and ensures that the security policies implemented by the TOE are always invoked.*	FPT_ITT.3 FPT_RCV.1 FPT_SEP.2 FPT_TRC_EXP.1	This objective requires the protection of the TSF (and its data) from external interference, tampering or inappropriate disclosure by mandating that the TSF create and maintain a domain for its execution. Domain is defined as the logical area that the TSF provides for itself in which to operate. Common mechanisms include hardware execution domains (e.g., processor execution rings as well as other isolation mechanisms that protect TSF data when it is in transit to other TSF components.) The requirements that implement this objective fall into two categories. The first category mandates mechanisms to implement a secure domain for execution. The second category mandates that if the TSF (for some reason) moves into an unknown or unconnected state, that it has a way to recover to a known or connected state. This ensures that the TSF can continue to protect itself even after unexpected interruptions. Requirements included in the first category are FPT_SEP.2, and FPT_ITT.3. FPT_SEP.2 was chosen to mandate mechanisms to disallow untrusted entities (executing in the context of TOE subjects) from modifying or interfering with the operation of the TSF. In addition to separating the TSF from untrusted subjects, the requirement also mandates that the cryptographic portion of the TSF be isolated from the rest of the TSF. Therefore, the cryptographic services are not only protected from untrusted subjects, but also from other code that is in the TSF that could potentially corrupt it. FPT_ITT.3 was chosen to protect TSF data in transmission between remote portions of the TSF and also requires that mechanisms be in place to protect against man-in-the-middle replay attacks which could attempt to interfere with the TSF policy being enforced. Requirements included in the second category are FPT_RCV.1 and FPT_TRP_EXP.1. FPT_RCV.1 is used to ensure that the TSF offers a mechanism to recover from a failed state by mandating that the TSF provide maintenance mode from which to re-initiate (or establish) a known (secure) state. This ensures that once the TSF has established a domain for its own execution it can always return to that state with confidence that this domain continues to be present. FPT_TRP_EXP.1 is used to address distributed TSFs and the fact that portions of these TSF may become disconnected over time. A disconnected portion of the TSF does not always suggest an insecure state or discontinuity of service (referenced in FPT_RCV.1). Instead, this requirement addresses the situation when a portion of a distributed TSF is disconnected from the rest of the TSF (with both

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		pieces continuing service). Specifically, it requires that there be mechanisms provided by the TSF to ensure that upon reconnection, the TSF portions will become in sync over a reasonable time period.
O.SECURE_STATE *The TOE will be able to verify the integrity of the TSF code and cryptographic data.*	FPT_TST.1(1) FPT_TST.1(2) FPT_TST_EXP.1 FPT_RCV.1	FPT_TST_EXP.1 ensures the correctness of the TSF software and TSF data. If TSF software is corrupted, it is possible that the TSF would no longer be able to enforce the security policies. This also holds true for TSF data. If TSF data is corrupt, the TOE may not correctly enforce its security policies. FPT_TST.1(1) and FPT_TST.1(2) address the critical nature and specific handling of the critical cryptographic security functions related to TSF data. Since the critical cryptographic security functions have specific requirements associated with them it is important to ensure that any fielded testing on the integrity of the data maintains the same level of security as specified in the functional requirements. FPT_RCV.1 ensures that the TOE does not continue to operate in an insecure state when a hardware or software failure occurs. Upon the failure of the TSF self-tests the TOE will enter a mode where it can no longer be assured of enforcing its security policies. Therefore, the TOE enters a state that disallows traffic flow and requires an administrator to follow documented procedures that instruct them on to return the TOE to a secure state. These procedures may include running diagnostics of the hardware, or utilities that may correct any integrity problems found with the TSF data or code. Solely specifying that the administrator reload and install the TOE software from scratch, while might be required in some cases, does not meet the intent of this requirement.
O.SOUND_DESIGN *The TOE will be designed using sound design principles and techniques. The TOE design, design principles and design techniques will be adequately and accurately documented.*	AVA_CCA_EXP.1 AVA_SOF.1 AVA_VLA.3 ADV_FSP_EXP.2 ADV_HLD_EXP.2 ADV_INT_EXP.1 ADV_LLD_EXP.1 ADV_RCR.1 ADV_SPM.1	Requirements for this objective fall into three categories. Category 1 requirements are those that mandate the developer provide detailed design information to the evaluators so that third party security analysis can be performed. Category 2 requirements mandate specific evaluator analysis be performed searching for vulnerabilities in the design. The result of Category 1 requirements feeds into the analysis required by Category 2 requirements. Category 3 requirements are those that mandate the developer to design the TOE in a modular fashion to minimize complexity of the TOE. Category 1 requirements comprise ADV_FSP_EXP.2, ADV_HLD_EXP.2, ADV_LLD_EXP.2, ADV_RCR.1 and ADV_SPM.1. In general, The ADV class of requirements is levied to aide in the understanding of the design for both parties, which ultimately helps to ensure the design is sound. ADV_FSP_EXP.2 requires that the interfaces to the TSF be completely specified. In this type of TOE, the full system call interface must be specified as well as any network protocols that are supported (should networking facilities be offered to untrusted users). Another interface that must not be overlooked (that often is) includes any trusted applications that are part of the TOE as well as the administrative interfaces. Having a complete understanding of what

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		is available at the TSF interface allows one to analyze this functionality in the context of design flaws. ADV_SPM.1 requires the developer to provide an informal model of the security policies of the TOE. Modeling these policies helps understand and reduce the unintended side-effects that occur during the TOE's operation that might adversely affect the TOE's ability to enforce its security policies. ADV_HLD_EXP.2 requires that a high-level design of the TOE be provided. This level of design describes the architecture of the TOE in terms of subsystems. It identifies which subsystems are responsible for making and enforcing security relevant (e.g., anything relating to a security functional requirement) decisions and provides a description, at a high level, of how those decisions are made and enforced. Having this level of description helps provide a general understanding of how the TOE works, without getting buried in details, and may allow the reader to discover flaws in the design. The low-level design, as required by ADV_LLD_EXP.1, provides the reader with the details of the TOE's design and describes at a module level how the design of the TOE addresses the security functional requirements. This level of description provides the detail of how modules interact within the TOE and if a flaw exists in the TOE's design, it is more likely to be found here rather than the high-level design. This requirement also mandates that the interfaces presented by modules be specified. Having knowledge of the parameters a module accepts, the errors that can be returned and a description of how the module works to support the security policies allows the design to be understood at its lowest level. The ADV_RCR.1 is used to ensure that the levels of decomposition of the TOE's design are consistent with one another. This is important, since design decisions that are analyzed and made at one level (e.g., functional specification) that are not correctly designed at a lower level may lead to a design flaw. This requirement helps in the design analysis to ensure design decisions are realized at all levels of the design. Category 2 requirements comprise AVA_CCA_EXP.1, AVA_VLA.3 and AVA_SOF.1. AVA_CCA_EXP.1 was created to require evaluators to perform analysis on potential cryptographic key leakage from the cryptographic module implemented by the TOE. Such analysis is important because key leakage is a grave vulnerability that could compromise information as it resides outside the scope of control of the TOE. In these cases, the cryptography is the only means of protection. Every effort must be taken during the course of the evaluation to identify problems with these functions. Because the protection of cryptographic keys are so important in ensuring the protection of TSF (and non TSF) data transmitted over untrusted medium, AVA_CCA_EXP.1 mandates that there be a thorough search for cryptographic key leakage from the cryptographic

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		module. AVA_VLA.3 requires the developer to perform a systematic search for potential vulnerabilities in all the TOE deliverables. For those vulnerabilities that are not eliminated, a rationale must be provided that describes why these vulnerabilities cannot be exploited by a threat agent with a moderate attack potential, which is in keeping with the desired assurance level of this TOE. As with the functional testing, a key element in this component is that an independent assessment of the completeness of the developer's analysis is made, and more importantly, an independent vulnerability analysis coupled with testing of the TOE is performed. This component provides the confidence that security flaws do not exist in the TOE that could be exploited by a threat agent of moderate (or lower) attack potential to violate the TOE's security policies. The AVA_SOF.1 requirement is applied to the user authentication mechanism. For this TOE, the strength of function specified is medium. This requirement ensures the developer has performed an analysis of the authentication mechanism to ensure the probability of guessing a user's authentication data would require a high-attack potential, as defined in Annex B of the CEM. Category 3 comprises only 1 requirement: ADV_INT_EXP.1. ADV_INT_EXP.1 ensures that the design of the TOE has been performed using good software engineering design principles that require a modular design of the TSF. Modular code increases the developer/evaluator understanding of the interactions within the TSF, which in turn, reduces the amount of errors in the design. Having a modular design is imperative for evaluator's to gain an appropriate level of understanding of the TOE's design in a relatively short amount of time.
O.SOUND _IMPLEMENTATION *The implementation of the TOE will be an accurate instantiation of its design.*	AVA_CCA_EXP.1 ALC_DVS.1 ALC_FLR.2 ATE_COV_EXP.1 ATE_DPT.2 ATE_FUN.1 ATE_IND.2 AVA_VLA.3 ADV_FSP_EXP.2 ADV_HLD_EXP.2 ADV_IMP_EXP.2 ADV_INT_EXP.1 ADV_LLD_EXP.1	These requirements combine to offer "analysis" based confidence (testing confidence is provided by O.FUNCTIONAL TESTING) that the TSF design is (and continues to be over time) translated error-free into an implementation. There are three categories of requirements that contribute to this objective: Category 1 requirements mandate a predictable and controlled development environment (to include process and procedures for flaw remediation); Category 2 requirements mandate that the source be documented in a manner that contributes to third party analysis and testing; and Category 3 requirements mandate actual vulnerability analysis and independent, third party analysis and testing to offer confidence that the TOE is implemented without flaws. Requirements that fall into Category 1 include: ALC_DVS.1, ALC_LCD.1, and ALC_TAT.1. ALC_DVS.1 requires the developer to describe the security measures they employ to ensure that the integrity and confidentiality of the TOE are maintained. The physical, procedural, and personnel security measures the developer uses

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
	ADV_RCR.1	provides an added level of control over who and how changes are made to the TOE and its associated evidence. ALC_FLR.2 plays a role in satisfying the "accurate instantiation" portion of this objective by requiring the developer to have procedures that address flaws that have been discovered in the product, either through developer actions (e.g., developer testing) or those discovered by others. The flaw remediation process used by the developer corrects any discovered flaws and performs an analysis to ensure new flaws are not created while fixing the discovered flaws. ALC_LCD.1 requires the developer to document the life-cycle model used in the development and maintenance of the TOE. This life-cycle model describes the procedural aspects regarding the development of the TOE, such as design methods, code or documentation reviews, and how changes to the TOE are reviewed and accepted or rejected. Requirements that fall into Category 2 include: ADV_FSP_EXP.2 ADV_HLD_EXP.2 ADV_IMP_EXP.2 ADV_INT_EXP.1 ADV_LLD_EXP.1, and ADV_RCR.1. These requirements contribute to O.SOUND_DESIGN but also contribute to this objective because they contribute to an evaluator's understanding of the TOE so that its implementation can be thoroughly tested. Of particular note in this category, is the role that ADV_INT_EXP.1 plays in meeting this objective. Where the other requirements mandate that the TOE be documented appropriately, ADV_INT_EXP.1 also requires the developer to not only design and document the TOE in a modular fashion, but to also "build" the TOE in a modular fashion (as implied by the term "good internal structure" in the ADV_INT_EXP.1.6D requirement). This requirement contributes heavily to the notion that if the system is built in a modular fashion, it will have fewer implementation flaws. See O.SOUND_DESIGN for an explanation of the rest of the requirements. Category 3 requirements include: AVA_VLA.3 AVA_CCA_EXP.1, and ATE_IND.1. To maintain consistency with the overall assurance goals of this TOE, O.SOUND_IMPLEMENTATION requires the AVA_VLA.3 component to provide the necessary level of confidence that vulnerabilities do not exist in the TOE that could cause the security policies to be violated. AVA_VLA.3 requires the developer to perform a systematic search for potential vulnerabilities in all the TOE deliverables. For those vulnerabilities that are not eliminated, a rationale must be provided that describes why these vulnerabilities cannot be exploited by a threat agent with a moderate attack potential, which is in keeping with the desired assurance level of this TOE. As with the functional testing, a key element in this component is that an independent assessment of the completeness of the developer's analysis is made, and more importantly, an independent vulnerability analysis coupled with testing of the TOE is performed.

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		ATE_IND.2 requires an independent confirmation of the developer's test results, by mandating a subset of the test suite be run by an independent party. This component also requires an independent party to attempt to craft functional tests that address functional behavior that is not demonstrated in the developer's test suite. Upon successful adherence to these requirements, the TOE's conformance to the specified security functional requirements will have been demonstrated. Because the protection of cryptographic keys are so important in ensuring the protection of TSF (and non TSF) data transmitted over unprotected medium, AVA_CCA_EXP.1 mandates that there be a thorough search only upon the cryptographic module for cryptographic key leakage in the TSF implementation.
O.TRAINED_USERS *The TOE will provide authorized users with the necessary guidance for secure use of the TOE, to include secure sharing of user data.*	AGD_USR.1	O.TRAINED_USERS requires that user's procedures for the secure use of the TOE be documented. AGD_USR.1 states that the developer shall provide user guidance describing the functions and interfaces available to the non-administrative users of the TOE and to users of the cryptographic module. The user guidance shall also describe the use of user-accessible security functions, and shall clearly present all user responsibilities necessary for secure operation of the TOE.
O.TRUSTED_PATH *The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data.*	FTP_TRP_EXP.1	FTP_TRP_EXP.1 requires the TOE to provide a mechanism that creates a distinct communication path that protects the data that traverses this path from disclosure or modification. This requirement ensures that an entity cannot insert itself between the user and the TOE. Since the user invokes the trusted path mechanism they can be assured they are communicating with the TOE. FTP_TRP_EXP.1 also mandates that the trusted path be the only means available for providing identification and authentication information, therefore ensuring a user's authentication data will not be compromised when performing authentication functions.
O.TSF_CRYPTOGRAPHIC_INTEGRITY *The TOE will provide cryptographic integrity mechanisms for TSF data while in transit to remote parts of the TOE.*	FPT_ITT.3	This objective requires the TOE to provide cryptography that must be used to protect TSF data as it is transmitted between parts of a physically distributed TOE. FPT_ITT.3 requires that the TSF shall be able to use encryption to detect modification, insertion and replay of TSF data transmitted between separate parts of the TOE.
O.USER_AUTHENTICATION *Users must authenticate their claimed identities (see O.USER_IDENTIFICATION) before they are allowed access to the TOE.*	FIA_SOS.1 FIA_UAU.1 FIA_UAU.6 FTA_SSL.1 FTA_SSL.2 FTP_TRP_EXP.1	FIA_UAU.1 plays a role in satisfying this objective by ensuring that every user is authenticated before the TOE performs any TSF-mediated actions on behalf of that user. FIA_UAU.6 ensures that the authorized user changing his authentication data re-authenticates before he or she is allowed to proceed. To verify the claimed identity of an authorized user, FIA_SOS.1 prescribes the metrics that must be satisfied. It provides the mechanism that will verify the secret for user authentication. The PP authors intentionally did not dictate that a password mechanism be

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		required and allowed for other types of authentication mechanisms (e.g. a PIN, Token). In any case, FIA_SOS.1 requires that the authentication mechanism provide the ability for authorized users to have a "secret" in a manner that cannot be guessed at random in less than one in 5 x 1015. FTA_SSL.1 and FTA_SSL.2 ensure that the authorized user authenticates him or herself before accessing a locked interactive session. This eliminates any chance for any user to acquire unauthorized access to an unattended session. Active interactive sessions may be locked by a user or after a specified time interval of user inactivity configured by an authorized administrator. FTP_TRP_EXP.1 requires the TOE to provide a mechanism that creates a distinct communication path that protects the data that traverses this path from disclosure or modification. This requirement ensures that the TOE can authenticate the end points and ensures that a user cannot insert themselves between the user and the TOE. It also mandates that the trusted path be the only means available for providing identification and authentication information during a TOE session establishment, for operations to modify authentication data, for protection of authentication data when a locked session is being unlocked and all other operations requiring a human user to enter authentication data, therefore ensuring a user's authentication data will not be compromised when performing authentication functions.
O.USER _IDENTIFICATION *The TOE will uniquely identify users.*	FIA_UID.1 FTP_TRP_EXP.1	FIA_UID.1 plays a role in satisfying this objective by ensuring that every user is identified before the TOE performs any TSF-mediated actions on behalf of that user. It also allows for the specification of a list of public objects that users are allowed read access before the user is identified. FTP_TRP_EXP.1 requires the TOE to provide a mechanism that creates a distinct communication path that protects the data that traverses this path from disclosure or modification. This requirement ensures that the TOE can identify the end points and ensures that a user cannot insert themselves between the user and the TOE. It also mandates that the trusted path be the only means available for providing identification and authentication information during a TOE session establishment, for operations to modify authentication data, for protection of authentication data when a locked session is being unlocked and all other operations requiring a human user to enter authentication data, therefore ensuring a user's authentication data will not be compromised when performing authentication functions.
O.VULNERABILITY _ANALYSIS *The TOE will undergo appropriate vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with moderate*	AVA_CCA_EXP.1 AVA_MSU.2 AVA_SOF.1 AVA_VLA.3	To maintain consistency with the overall assurance goals of this TOE, AVA_VLA.3 component is required to provide the necessary level of confidence that vulnerabilities do not exist in the TOE that could cause the security policies to be violated. AVA_VLA.3 requires the developer to perform a systematic search for potential vulnerabilities in all the TOE deliverables. For those vulnerabilities that are not eliminated, a rationale must be provided that describes why these vulnerabilities cannot be exploited by a threat agent with a moderate

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
attack potential to violate the TOE's security policies.		attack potential, which is in keeping with the desired assurance level of this TOE. As with the functional testing, a key element in this component is that an independent assessment of the completeness of the developer's analysis is made, and more importantly, an independent vulnerability analysis coupled with testing of the TOE is performed. This component provides the confidence that security flaws do not exist in the TOE that could be exploited by a threat agent of moderate attack potential to violate the TOE's security policies. AVA_CCA_EXP.1 requires that evaluators perform analysis on potential cryptographic key leakage from the cryptographic module implemented by the TOE. Such analysis is important because key leakage is a grave vulnerability that could compromise information while TSF (and non TSF) data is transmitted over an unprotected medium. In these cases, the cryptography is the only means of protection. Every effort must be taken during the course of the evaluation to identify problems with these functions. AVA_MSU.2 ensures that an analysis for any vulnerability that might be caused by unclear documentation is performed. Conflicting, misleading, incomplete or unreasonable guidance may result in a user of the TOE believing that the TOE is secure when it is not, and can result in vulnerabilities. This guidance lists all assumptions about the intended environment and all requirements for external security measures (including external procedural, physical and personnel controls). AVA_SOF.1 ensures that an analysis of the strength of the functions is performed. Even if a TOE security function cannot be bypassed, deactivated, or corrupted, it may still be possible to defeat it because there is a vulnerability in the concept of its underlying security mechanisms. For those functions a qualification of their security behavior can be made using the results of a quantitative or statistical analysis of the security behavior of these mechanisms and the effort required to overcome them. The qualification is made in the form of a strength of TOE security function claim.

7.5 Explicit Requirements Rationale

249 Explicit components have been included in this protection profile because the Common Criteria requirements were found to be insufficient as stated. Tables 7.5 and 7.6 include the rationale for using explicit requirements.

7.5.1 Explicit Functional Requirements

Table 7.5 – Rationale for Explicit Functional Requirements

Explicit Component	Rationale

Explicit Component	Rationale
FCS_BCM_EXP.1	The CC does not provide a means of specifying a cryptographic module baseline for implementations developed in hardware, in software, or in hardware/software combinations. FCS_BCM_EXP.1 provides for the specification of the required FIPS certification based on the implementation baseline.
FCS_CKM_EXP.1	The CC cryptographic support section does not specifically address the concepts of key validation techniques and key packaging. Although closely tied to generated keys, these concepts typically get implemented after, not during, the actual generation of a key. In this PP, FCS_CKM_EXP.1 allows for specifically addressing these key management-related concepts.
FCS_CKM_EXP.2	The CC does not provide components for key handling and storage. Key access and key destruction components do not address keys being transferred within the device nor key archiving when key is not in use. FCS_CKM_EXP.2 addresses internal key transfer and archiving. It also addresses the handling of storage areas where keys reside.
FCS_COA_EXP.1	The CC FCS families address the management of cryptographic keys and the operational use of those cryptographic keys to help satisfy several high-level security objectives. Another reason for having the cryptographic functionality in the TOE is for applications to be able to utilize the cryptographic operations. FCS_COA_EXP.1 was created to require a means for applications to be able to utilize the cryptographic functionality contained in the TOE.
FCS_COP_EXP.1	The CC cryptographic operation components are focused on specific algorithm types and operations requiring specific key sizes. The generation of random numbers can be better stated as an explicit component. Neither algorithms nor keys are required to generate random numbers. Random number generators can use any combination of software-based or hardware-based inputs as long as the RNG/PRNG design requirements are met and the required RNG/PRNG tests are successful.

Explicit Component	Rationale
FPT_TRC_EXP.1	FPT_TRC_EXP has been created to require timely consistency of replicated TSF data. Although there is a Common Criteria Requirement that attempts to address this functionality, it falls short of the needs of the environment in this protection profile. Specifically, FPT_TRC.1.1 states that "The TSF shall ensure that TSF data is consistent when replicated between parts of the TOE." In the widely distributed environment of this PP's TOE, this is an infeasible requirement. For TOEs with a very large number of components, 100 percent TSF data consistency is not achievable and is not expected at any specific instant in time. Another concern lies in FPT_TRC.1.2 which states that when replicated parts of the TSF are "disconnected", the TSF shall ensure consistency of the TSF replicated data upon "reconnection". Upon first inspection, this seems reasonable, however, when applying this requirement it becomes clear that it dictates specific mechanisms to determine when a component is "disconnected" from the rest of the TSF and when it is "reconnected". This is problematic in this PP's environment in that it is not the intent of the authors to dictate that distributed TSF components keep track of connected/disconnected components. In general, to meet the needs of this PP, it is acceptable to simply require a mechanism that provides TSF data consistency in a timely manner after it is determined that it is inconsistent.
FPT_TST_EXP.1	FPT_TST_EXP.1 has been created because the FPT_TST.1.2 element was removed from the original component FPT_TST.1. The element FPT_TST.1.2 states that TSF shall provide authorized users with the capability to verify the integrity of TSF data. This not a feasible requirement. Verifying the integrity of TSF data (e.g., passwords, session keys) is not feasible because it is constantly being updated.
FTP_TRP_EXP.1	FTP_TRP_EXP.1 has been created because the original FTP_TRP.1.1 element was changed and in a sense weakened from a security perspective. The original CC element states that the TSF shall provide assured identification between end points (i.e., a two way assured identification) and the explicit element FTP_TRP_EXP.1.1 only requires assured identification between the TSF to the user (i.e., a one way assured identification).

7.5.2 Explicit Assurance Requirements

Table 7.6 – Rationale for Explicit Assurance Requirements

Explicit Component	Rationale
ADV_ARC_EXP.1	See Appendix E
ADV_FSP_EXP.1	See Appendix E
ADV_HLD_EXP.1	See Appendix E

Explicit Component	Rationale
ADV_IMP_EXP.2	The intent of the PP authors is to require the implementation representation only between a subset of all portions of the implementation. The ADV_IMP.2 requires that the implementation representation be describe the relationships between all portions of the implementation. The CC does not have a component that matches the authors intent for this explicit component.
ADV_INT_EXP.1	See Appendix E
ADV_LLD_EXP.1	See Appendix E
AMA_AMP_EXP.1	AMA_AMP_EXP.1 has been created to ensure that the TOE, once it has been evaluated and found compliant with this protection profile, will undergo continual security analysis of future enhancements and modifications to ensure that compliance with the PP is maintained. AMA_AMP_EXP.1 requires the TOE developer to provide a plan ensuring that the evaluated level of assurance will be maintained. The plan must describe all processes and procedures that will define how the TOE security analyst(s) will be an integral component in the analysis and approval of changes to the TOE to ensure that security issues are appropriately addressed. The plan must also identify the required technical qualifications of the TOE security analyst and how their technical competency will be maintained.
ATE_COV_EXP.2	The intent of the PP authors is for an NSA evaluator to confirm that the information provided for cryptographic portions of the TOE meets all requirements for content and presentation of evidence. ATE_COV_EXP.2 is exactly the same as ATE_COV.2 except for the addition of the last element ATE_COV_EXP.2.2E which states what's in the previous paragraph.
AVA_CCA_EXP.1	The intent of the PP authors is to require covert channel analysis only on the cryptographic module(s) and the CC does not have requirements to perform partial covert channel analysis on TOE. AVA_CCA_EXP.1 provides covert channel analysis only upon the cryptographic module in search for leakage of critical cryptographic security parameters.

7.6 Rational for Strength of Function

291 The TOE minimum strength of function is SOF-medium. The evaluated TOE is intended to operate in DoD medium robustness environments processing up to U.S. Government classified information. The minimum strength of function was chosen to be consistent with FIA_SOS.1 by providing a probability of successful authentication for a random attempt of less than one in 5 x 10^{15}. This security function is in turn consistent with the security objectives described in section 7.4.

292 The minimum SOF does not apply to any cryptographic mechanisms with respect to a CC evaluation. The strength of cryptographic algorithms is outside the scope of the CC. The strength of the cryptographic mechanisms will be determined by NIST FIPS 140-2 certified modules and requirements specified in this PP; the validation of these cryptographic mechanisms will be performed by the NSA.

7.7 Rationale for Assurance Rating

293 This protection profile has been developed for a U.S. Government medium robustness environment. The TOE environment and the value of information processed by this environment (i.e., single-level and system high) establish the need for the TOE to be evaluated at an Evaluated Assurance Level 4 Augmented (EAL4+)[22]. This protection profile is consistent with EAL4+.

[22] Refer to the "Mutual Recognition of Common Criteria Certificates" section 1.3 to read conditions for the CC certificate to be mutually recognized for PPs with EALs higher than 4.

8. References

[1] Common Criteria Implementation Board, Common Criteria for Information Technology Security Evaluation, CCMB-2005-08-01, Version 2.3, August 2005

[2] Department of Defense Chief Information Officer, Guidance and Policy for Department of Defense Information Assurance Memorandum No. 6-8510 dated 16 June 2000

[3] National Institute of Standards and Technology, Data Encryption Standard (DES); specifies the use of Triple DES, Federal Information Processing Standard Publication (FIPS-PUB) 46-3, dated October 1999.

[4] National Institute of Standards and Technology, Digital Signature Standard (DSS), Federal Information Processing Standard Publication (FIPS-PUB) 186-2, dated January 2000.

[5] National Institute of Standards and Technology, Key Management Using ANSI X9.17, Federal Information Processing Standard Publication (FIPS-PUB) 171, dated April 1992.

[6] National Institute of Standards and Technology, Secure Hash Standard, Federal Information Processing Standard Publication (FIPS-PUB) 180-1, dated April 1995.

[7] National Institute of Standards and Technology, Secure Hash Standard, Federal Information Processing Standard Publication (FIPS-PUB) 180-2, dated 1 August 2002.

[8] National Institute of Standards and Technology, Security Requirements for Cryptographic Modules, Federal Information Processing Standard Publication (FIPS-PUB) 140-1, dated January 11, 1994.

[9] National Institute of Standards and Technology, Security Requirements for Cryptographic Modules, Federal Information Processing Standard Publication (FIPS-PUB) 140-2, dated May 25, 2001.

[10] National Security Agency, Controlled Access Protection Profile (CAPP), Version 1.d, 8 October 1999

[11] National Security Agency, Information Assurance Technical Framework (IATF), Version 2.0.1 - September 1999.

[12] Department of Defense Standard, Department of Defense Trusted Computer System Evaluation Criteria (Orange Book), December 1985

[13] Trusted Product Evaluation Program (TPEP) Trusted Computer System Evaluation Criteria (TCSEC) Interpretations

[14] American National Standards Institute (ANSI) X9.42-2001, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography.

[15] American National Standards Institute (ANSI) X9.44-2000, Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport using Factoring-Based Cryptography.

[16] American National Standards Institute (ANSI) X9.31-1998 (May 1998), Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA).

[17] American National Standards Institute (ANSI) X9.63-200x (1 Oct 2000), Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography.

[18] American National Standards Institute (ANSI) X9.80 (3 January 2000), Prime Number Generation, Primality Testing, and Primality Certificates.

[19] American National Standards Institute (ANSI) X9.17-1985, Financial Institution Key Management (Wholesale).

[20] American National Standards Institute (ANSI) X9.52-1998, Triple Data Encryption Algorithm Modes of Operation.

[21] American National Standards Institute (ANSI) X9.62-1-xxxx (10 Oct 1999), Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm (ECDSA).

[22] National Institute of Standards and Technology Special Publication 800-22: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, October 2000.

[23] National Institute of Standards and Technology Special Publication 800-56: Recommendation On Key Establishment Schemes, Draft 2.0, January 2003.

[24] Public-Key Cryptography Standards, PKCS #5: Password-Based Encryption Standard, Version 2.0, dated March 25, 1999.

[25] Public-Key Cryptography Standards, PKCS #8: Private-Key Information Syntax Standard, Version 1.2.

[26] Public-Key Cryptography Standards, PKCS #11: Cryptographic Token Interface Standard, Version 2.11.

[27] Information Technology Management Reform Act of 1996, 40 U.S.C. Section 5142,

Appendix A - Acronyms

ANSI	American National Standards Institute
CC	Common Criteria for Information Technology Security Evaluation Version 2.3
COTS	Commercial-Off-The-Shelf
DAC	Discretionary Access Control
DoD	Department of Defense
EAL	Evaluation Assurance Level
FIPS	Federal Information Processing Standard
IA	Information Assurance
IT	Information Technology
NIST	National Institute of Standards and Technology
OS	Operating System
PKCS	Public Key Cryptography Standards
PKI	Public Key Infrastructure
PP	Protection Profile
RNG	Random Number Generator
SF	Security Function
SFP	Security Function Policy
SFR	Security Function Requirement
SOF	Strength of Function
ST	Security Target
TOE	Target of Evaluation
TOM	Target of Maintenance
TSC	TSF Scope of Control
TSF	TOE Security Functions
TSFI	TSF Interface
TSP	TOE Security Policy

Appendix B - Cryptographic Standards, Policies, and Other Publications

Standards

ANSI X9.17	American National Standards Institute (ANSI) X9.17-1985, Financial Institution Key Management (Wholesale), [http://webstore.ansi.org/ansidocstore/find.asp].
ANSI X9.31-1998	American National Standards Institute (ANSI) X9.31-1998 (May 1998), Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA), [http://webstore.ansi.org/ansidocstore/find.asp].
ANSI X9.42-2001	American National Standards Institute (ANSI) X9.42-2001, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography, [http://webstore.ansi.org/ansidocstore/find.asp].
ANSI X9.52-1998	American National Standards Institute (ANSI) X9.52-1998, Triple Data Encryption Algorithm Modes of Operation, [http://webstore.ansi.org/ansidocstore/find.asp].
ANSI X9.62-1	American National Standards Institute (ANSI) X9.62-1-xxxx (10 Oct 1999), Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm (ECDSA), (http://webstore.ansi.org/ansidocstore/find.asp).
ANSI X9.63	American National Standards Institute (ANSI) X9.63-200x (1 Oct 2000), Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography, [http://webstore.ansi.org/ansidocstore/find.asp].
ANSI X9.80	American National Standards Institute (ANSI) X9.80 (3 January 2000), Prime Number Generation, Primality Testing, and Primality Certificates, [http://webstore.ansi.org/ansidocstore/find.asp].
FIPS PUB 46-3	National Institute of Standards and Technology, Data Encryption Standard (DES); specifies the use of Triple DES, Federal Information Processing Standard Publication (FIPS-PUB) 46-3, dated October 1999, [http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
FIPS PUB 140-2	National Institute of Standards and Technology, Security Requirements for Cryptographic Modules, Federal Information Processing Standard Publication (FIPS-PUB) 140-2, dated May 25, 2001, [http://cs-www.ncsl.nist.gov/publications/fips/fips140-2/fips1402.pdf].
FIPS PUB 171	National Institute of Standards and Technology, Key Management Using ANSI X9.17, Federal Information Processing Standard Publication (FIPS-PUB) 171, dated April 1992 , updated February 2001[http://csrc.nist.gov/cryptval/des/kmvsval.htm].
FIPS PUB 180-2	National Institute of Standards and Technology, Secure Hash Standard, Federal Information Processing Standard Publication (FIPS-PUB) 180-2, dated 1 August 2002, [http://cs-www.ncsl.nist.gov/publications/fips/fips180-2/fips180-2.pdf].

FIPS PUB 186-2 National Institute of Standards and Technology, Digital Signature Standard (DSS), Federal Information Processing Standard Publication (FIPS-PUB) 186-2, dated January 2000 [http://cs-www.ncsl.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf].

FIPS PUB 197 National Institute of Standards and Technology, Advanced Encryption Standard, Federal Information Processing Standard Publication (FIPS-PUB) 197, dated November 2001, [http://cs-www.ncsl.nist.gov/publications/fips/fips197/fips-197.pdf].

PKCS#5 Vers.2.0 Public-Key Cryptography Standards, PKCS #5: Password-Based Encryption Standard, Version 2.0, dated March 25, 1999, [http://www.rsasecurity.com/rsalabs/pkcs/pkcs-5/index.html].

PKCS#8 Vers.1.2 Public-Key Cryptography Standards, PKCS #8: Private-Key Information Syntax Standard, Version 1.2, [http://www.rsasecurity.com/rsalabs/pkcs/pkcs-8/index.html].

PKCS#11 Vers.2.11 Public-Key Cryptography Standards, PKCS #11: Cryptographic Token Interface Standard, Version 2.11, [http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/index.html].

PKCS#12 Vers.1.0 Public-Key Cryptography Standards, PKCS #12: Personal Information Exchange Syntax Standard, Version 1.0, [http://www.rsasecurity.com/rsalabs/pkcs/pkcs-12/index.html].

Policies

Certificate Policy X.509 Certificate Policy for the United States Department of Defense, Version 7.0, 18 December 2002, [http://iase.disa.mil/policy.html#pki].

Other Publications

NIST S.P. 800-22 National Institute of Standards and Technology Special Publication 800-22: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, October 2000, [http://cs-www.ncsl.nist.gov/publications/nistpubs/800-22/sp-800-22-051501.pdf].

NIST S.P. 800-56 National Institute of Standards and Technology Special Publication 800-56: Recommendation On Key Establishment Schemes, Draft 2.0, January 2003, [http://csrc.nist.gov/CryptoToolkit/kms/keyschemes-Jan03.pdf].

NIST S.P. 800-38A National Institute of Standards and Technology Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation - Methods and Techniques, December 2001, [http://cs-www.ncsl.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf].

PKI Roadmap Public Key Infrastructure Roadmap for the Department of Defense, Version 5.0, 18 December 2000, [http://iase.disa.mil/pki/roadmap.html].

Appendix C - Statistical Random Number Generator Tests

294 A cryptographic module employing random number generators (RNGs) shall perform the following statistical tests for randomness. A single bit stream of 20,000 consecutive bits of output from each RNG shall be subjected to the following four tests: monobit test, poker test, runs test, and long runs test. (These four tests are simply those that formerly existed as the statistical RNG tests in Federal Information Processing Standard 140-2. However, for purposes of meeting this protection profile, these tests must be performed at the frequency specified earlier in this protection profile.)

295 The Monobit Test:

1. Count the number of ones in the 20,000 bit stream. Denote this quantity by X.
2. The test is passed if $9{,}725 < X < 10{,}275$.

296 The Poker Test:

1. Divide the 20,000 bit stream into 5,000 contiguous 4 bit segments. Count and store the number of occurrences of the 16 possible 4 bit values. Denote *f(i)* as the number of each 4 bit value *i*, where $0 \leq i \leq 15$.
2. Evaluate the following:

$$X = (16 / 5000) * \left(\sum_{i=0}^{15} [f(i)]^2 \right) - 5000$$

3. The test is passed if $2.16 < X < 46.17$.

297 The Runs Test:

1. A run is defined as a maximal sequence of consecutive bits of either all ones or all zeros that is part of the 20,000 bit sample stream. The incidences of runs (for both consecutive zeros and consecutive ones) of all lengths (≥ 1) in the sample stream should be counted and stored.

2. The test is passed if the runs that occur (of lengths 1 through 6) are each within the corresponding interval specified in the table below. This must hold for both the zeros and ones (i.e., all 12 counts must lie in the specified interval). For the purposes of this test, runs of greater than 6 are considered to be of length 6.

Table C.1 - Required Intervals for Length of Runs Test

Length of Run	Required Interval
1	2343 - 2657
2	1135 - 1365
3	542 - 708
4	251 - 373
5	111 - 201
6 and greater	111 - 201

298 The Long Runs Test:

1. A long run is defined to be a run of length 26 or more (of either zeros or ones).
2. On the sample of 20,000 bits, the test is passed if there are no long runs.

Appendix D - Randomizer Qualification Testing Requirements

299 This test utilizes the NIST battery of statistical tests as described in "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", NIST Special Publication 800-22. This document and corresponding software code are available for downloading at the following Internet sites: http://csrc.ncsl.nist.gov/rng or http://csrc.ncsl.nist.gov/CryptoToolkit/tkrng .

300 The Randomizer Qualification Statistical Test Suite consists of the following statistical tests:

1. Frequency (Monobit) Test
2. Frequency Test within a Block
3. Cumulative Suns (Cusum) Test
4. Runs Test
5. Longest Run of ones in a Block
6. Binary Matrix Rank Test
7. Discrete Fourier Transform (Spectral) Test
8. Maurer's Universal Statistical Test
9. Approximate Entropy Test
10. Serial Test

Randomizer Qualification Test Process

301 Power up the randomizer and collect a sample of 100,000 bits of data every 5 minutes until 10 samples have been collected. Concatenate the 10 samples to form a single sample of length 1,000,000 bits. Apply the above statistical tests using the following input parameters:

Sequence Length: 100,000

Number of Sequences: 10

Block Frequency Test Block Length: 100

Universal Test Block Length: 6

Universal Test Number of Initialization Steps: 640

Approximate Entropy Block Length: 10

Serial Test Block Length: 10

302 Each statistical test will produce a series of 10 P-Values. The Cusum and Serial test consist of two tests each and produces two series of 10 P-Values each. Thus the statistical test suite will produce twelve series of 10 P-Values each. The collected sample of data passes the statistical test suite if for each of the twelve series of P-Values at least 9 of the 10 P-Values are greater than 0.01. The NIST software generates a file, finalAnalysisReport, which summarizes the results of the tests. The data passes the statistical test suite if all of the twelve values listed in the proportions column are greater than or equal to 0.9.

303 The above test procedure is to be repeated 3 times. The randomizer passes the randomizer qualification test if the statistical test suite is passes on at least 2 of the 3 attempts.

Appendix E - Rationale for Explicit ADV Assurance Requirements

E.1 Rationale for ADV_ARC_EXP.1

Objectives

304 The architectural design of the TOE is related to the information contained in other decomposition documentation (functional specification, high-level design, low-level design) provided for the TSF, but presents the design in a manner that supports the argument that the TSP cannot be compromised (FPT_SEP) and that it cannot be bypassed (FPT_RVM). The objective of this component is for the developer to provide an architectural design and justification associated with the integrity and non-bypassability properties of the TSF.

Application notes

305 FPT_SEP and FPT_RVM are distinct from other SFRs because they largely have no directly observable interface at the TSF. Rather, they are properties of the TSF that are achieved through the design of the system, and enforced by the correct implementation of that design. Because of their pervasive nature, the material needed to provide the assurance that these requirements are being achieved is better suited to a presentation separate from the design decomposition of the TSF as embodied in ADV_FSP_EXP, ADV_HLD_EXP, and ADV_LLD_EXP. This is not to imply that the architectural design called for by this component cannot reference or make use of the design composition material; but it is likely that much of the detail present in the decomposition documentation will not be relevant to the argument being provided for the architectural design document.

306 The architectural design document consists of two types of information. The first is the design information for the entire TSF related to the FPT_SEP and FPT_RVM requirements. This type of information, like the decompositions for ADV_HLD_EXP and ADV_FSP_EXP, describes *how* the TSF is implemented. The description, however, should be focused on providing information sufficient for the reader to determine that the TSF implementation is likely not to be compromised, and that the TSP enforcement mechanisms (that is, those that are implementing SFRs other than FPT_SEP and FPT_RVM) are likely always being invoked.

307 The nature of the FPT_SEP requirement lends itself to a design description much better than FPT_RVM. For FPT_SEP, mechanisms can be identified (e.g., memory management, protected processing modes provided by the hardware, etc.) and described that implement the domain separation.

308 However, FPT_RVM is concerned with interfaces that may bypass the enforcement mechanisms. In most cases this is a consequence of the implementation, where if a programmer is writing an interface that accesses or manipulates an object, it is that programmer's responsibility to use interfaces that are part of the TSP enforcement mechanism for the object and not to try to "go around" those interfaces. However, the developer is still able to describe architectural elements

(e.g., object managers, macros to be invoked for specific functionality) that pertain to the design of the system to achieve the "always invoked" property of the TSF.

309 For FPT_SEP, the design description should cover how user input is handled by privileged-mode routines; what hardware self-protection mechanisms are used and how they work (e.g., memory management hardware, including translation lookaside buffers); how software portions of the TSF use the hardware self-protection mechanisms in providing their functions; and any software protection constructs or coding conventions that contribute to meeting FPT_SEP.

310 For FPT_RVM, the description should cover resources that are protected under the SFPs (usually FDP_* components) and other security relevant functionality (e.g., audit) that is provided by the TSF. The description should also identify the interfaces that are associated with each of the resources or the functionality; this might make use of the information in the FSP. This description should also describe any design constructs, such as object managers, and their method of use. For instance, if routines are to use a standard macro to produce an audit record, this convention is a part of the design that contributes to the non-bypassability of the audit mechanism. It's important to note that "non-bypassability" in this context is not an attempt to answer the question "could a part of the TSF implementation, if malicious, bypass a TSP mechanism", but rather it's to document how the actual implementation does not bypass the mechanisms implementing the TSP.

311 In addition to the descriptive information indicated in the previous paragraphs, the second type of information an architectural design document must contain is a justification that the FPT_SEP and FPT_RVM requirements are being met. This is distinct from the description, and presents an argument for why the design presented in the description is sufficient.

312 For FPT_SEP, the justification should cover the possible modes by which the TSF could be compromised, and how the mechanisms implemented in response to FPT_SEP counter such compromises. The vulnerability analysis might be referenced in this section.

313 For FPT_RVM, the justification demonstrates that whenever a resource protected by an SFR is accessed, the protection mechanisms of the TSF are invoked (that is, there are no "backdoor" methods of accessing resources that are not identified and analyzed as part of the ADV_FSP_EXP/ADV_HLD_EXP/ADV_LLD_EXP analysis). Similarly, the description demonstrates that a function described by an SFR is always provided where required. For example, if the FCO_NRO family were being used the description should demonstrate that all interfaces either 1) do not deal with transmitting the information identified in the FCO_NRO component included in the ST, or 2) invoke the mechanism(s) described by the decomposition documentation. The justification for FPT_RVM will likely need to address all of the TSFI in order to make the case that the TSP is non-bypassable.

E.2 Rationale for ADV_FSP_EXP.1

Objectives

314 The functional specification is a description of the user-visible interface to the TSF. It contains an instantiation of the TOE security functional requirements. The functional specification must include all of the user-visible TOE security functional requirements.

315 An interface exists at the TSF boundary if it can be used by an administrator; untrusted user or program; or another TOE. The requirements in this family apply to all types of TSFI, not just APIs.

Application notes

316 A description of the TSF interfaces (TSFI) provides fundamental evidence on which assurance in the TOE can be built. The functional specification provides a description of *what* the TSF provides to users (as opposed to the high-level design and low-level design, which provide a description of *how* the functionality is provided). Further, the functional specification provides this information in the form of interface (TSFI) documentation.

317 In order to identify the software interfaces to the TSF, the subsystems of the TOE that make up the TSF must be identified. This identification is required by ADV_HLD_EXP. A portion of the TOE is considered to be in the TSF under two conditions:

a) The software contributes to the satisfaction of security functionality specified by a functional requirement in the ST. This is typically all software that runs in a privileged state of the underlying hardware, as well as software that runs in unprivileged states that performs security functionality.

b) The software used by administrators in order to perform security management activities specified in the guidance documentation. These activities are a superset of those specified by any FMT_* functional requirements in the ST.

318 Identification of the TSFI is a complex undertaking. The TSF provides services and resources, and the TSFI are the interfaces *to* those security services/resources.

319 The TSF may have dependencies on the IT environment *using* services of the IT environment. While these are (using the general term) interfaces between the TSF and the IT environment, they are not TSFI. Nonetheless, it is vital to document their existence to integrators and consumers of the system, and thus documentation requirements for these interfaces are specified in ADV_ING.

320 Having discussed the interfaces in general, the types of TSFI are now discussed in more detail. This discussion categorizes the TSFI into the two categories mentioned previously: TSFI to software directly implementing the SFRs, and TSFI used by administrators.

321 TSFI in the first category are varied in their appearance in a TOE. Most commonly interfaces are thought of as those described in terms of Application Programming Interfaces (APIs), such as kernel calls in a Unix-like operating system. However, interfaces also may be described in terms of menu choices, check boxes, and edit boxes in a GUI; parameter files (the *.INI files and the registry for Microsoft Windows systems); and network communication protocols at all levels of the protocol stack.

322 TSFI in the second category are more complex. While there are three cases that need to be considered (discussed below), for all cases there is an “additional” requirement that the functions that an administrator uses to perform their duties—as documented in administrative guidance—also are part of the TSFI and must be documented and shown to work correctly. The individual cases are as follows:

a) The administrative tool used is also accessible to untrusted users, and runs with some "privilege" itself. In this case the TSFI to be described are similar to those in the first category because the tool itself is privileged.

b) The administrative tool uses the privileges of the invoker to perform its tasks. In this case, the interfaces supporting the activities that the administrator is directed to do by the administrative guidance (AGD_ADM, including FMT_* actions) are part of the TSFI. Note that this case differs from the previous one in that the tool does not run with privilege, and therefore is not in and of itself interesting from a security point of view. Also note that when FPT_SEP is included in the ST, the executable image of such tools need to be protected so that an untrusted user cannot replace the tool with a "trojan" tool.

c) The administrative tool is only accessible to administrative users. In this case the TSFI are identified in the same manner as the previous case. Unlike the previous case, however, the evaluator ascertains that an untrusted user is unable to invoke the tool when FPT_SEP is included in the ST.

323 All TSFI are *security relevant*, but some interfaces (or aspects of interfaces) are more critical and require more analysis than other interfaces. If an interface plays a role in enforcing any security policy on the system, then that interface is *security enforcing*. Such policies are not limited to the access control policies, but also refer to any functionality provided by one of the SFRs contained in the ST (with exceptions for FPT_SEP and FPT_RVM as detailed below). Note that it is possible that an interface may have various effects and exceptions, some of which may be security enforcing and some of which may not.

324 FPT_SEP and FPT_RVM are SFRs that require a different type of analysis from other SFRs. These requirements are architecturally related, and their implementation (or lack thereof) is not easily (or efficiently) testable at the TSFI. From a terminology standpoint, although implementation (and the associated analysis) of FPT_SEP and FPT_RVM is critical to the trustworthiness of the system, these two SFRs will not be considered as SFRs that are applicable when determining the set of security-enforcing TSFIs as defined in the previous paragraph.

325 Interfaces (or parts of an interface) that need only to function correctly in order for the security policies of the system to be preserved are termed *security supporting*. A security supporting interface typically plays a role in supporting the architectural requirements (FPT_SEP or FPT_RVM), meaning that as long as it can be shown that it does not allow the TSF to be compromised or bypassed no further analysis against SFRs is required. In order for an interface to be security supporting it must have *no* security enforcing aspects. In contrast, a security enforcing interface may have security supporting aspects (for example, the ability to set the system clock may be a security enforcing aspect of an interface, but if that same interface is used to display the system date that effect may only be security supporting).

326 A key aspect for the assurance associated with this component is the concept of the evaluator being able to verify that the developer has correctly categorized the security enforcing and security supporting interfaces. The requirements are structured such that the information required for security supporting interfaces is the *minimum* necessary in order for the evaluator to make this determination in an effective manner.

327 For the purposes of the requirements, interfaces are specified (in varying degrees of detail) in terms of their parameters, parameter descriptions, effects, exceptions, and error messages. Additionally, the purpose of each interface, and the way in which the interface is used (both from the point of view of the external stimulus (e.g., the programmer calling the API, the administrator changing a setting in the registry) and the effect on the TSFI that stimulus has) must be specified. This description of method of use must also specify how those administrative interfaces that are unable to be successfully invoked by untrusted users (case "c" mentioned above) are protected.

328 Parameters are explicit inputs to and outputs from an interface that control the behavior of that interface. For examples, parameters are the arguments supplied to an API; the various fields in a packet for a given network protocol; the individual key values in the Windows Registry; the signals across a set of pins on a chip; etc.

329 A parameter description tells what the parameter is in some meaningful way. For instance, the interface "foo(i)" could be described as having "parameter i which is an integer"; this is not an acceptable parameter description. A description such as "parameter i is an integer that indicates the number of users currently logged in to the system." is required.

330 Effects of an interface describe what the interface does. The effects that need to be described in an FSP are those that are visible at any external interface, not necessarily limited to the one being specified. For instance, the sole effect of an API call is not just the error code it returns. Also, depending on the parameters of an interface, there may be many different effects (for instance, an API might have the first parameter be a "subcommand", and the following parameters be specific to that subcommand. The IOCTL API in some Unix systems is an example of such an interface).

331 Exceptions refer to the processing associated with "special checks" that may be performed by an interface. An example would be an interface that has a certain set of effects for all users except the Superuser; this would be an exception to the normal effect of the interface. Use of a privilege for some kind of special effect would also be covered in this topic.

332 Documenting the errors associated with the TSF is not as straight-forward as it might appear, and deserves some discussion. A general principle is that errors generated by the TSF that are visible to the user should be documented. These errors can be the direct result of invoking a TSFI (an API call that returns an error); an indirect error that is easily tied to a TSFI (setting a parameter in a configuration that is error-checked when read, returning an immediate notification); or an indirect error that is not easily tied to a TSFI (setting a parameter that, in combination with certain system states, generates an error condition that occurs at a later time. An example might be resource exhaustion of a TSF resource due to setting a parameter to too low of a value).

333 Errors can take many forms, depending on the interface being described. For an API, the interface itself may return an error code; set a global error condition, or set a certain parameter with an error code. For a configuration file, an incorrectly configured parameter may cause an error message to be written to a log file. For a hardware PCI card, an error condition may raise a signal on the bus, or trigger an exception condition to the CPU.

334 For the purposes of the requirements, errors are divided into two categories. The first category includes *direct errors*, which are directly related to a TSFI; examples are API calls and

parameter-checking for configuration files. For this category of errors, the functional specification must document all of the errors that can be returned as a result of invoking a security-enforcing aspect of the interface such that a reader should be able to associate an interface with the errors it is capable of generating. The second category includes *indirect errors*, which are errors that are not directly tied to the invocation of a TSFI, but which are reported to the user as a result of processing that occurs in the TSF. It should be noted that while the condition that causes the indirect error can be documented; it is generally much harder to document all the ways in which that condition can occur.[23] Because of the difficulty associated with documenting all of the ways to cause an error, and because of the cost of documenting all indirect errors compared to the benefit of having them documented, indirect errors are not required to be documented.

335 The ADV_FSP_EXP.1.2E element defines a requirement that the evaluator determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the functional specification, in addition to the pairwise correspondences required by the ADV_RCR family. Although the evaluator may use the evidence provided in ADV_RCR as an input to making this determination, ADV_RCR cannot be the basis for a positive finding in this area. The requirement for completeness is intended to be relative to the level of abstraction of the functional specification.

E.3 Rationale for ADV_HLD_EXP.1

Objectives

336 The high-level design of a TOE provides both context for a description of the TSF, and a thorough description of the TSF in terms of major structural units (i.e. subsystems). It relates these units to the functions that they provide. The high-level design requirements are intended to provide assurance that the TOE provides an architecture appropriate to implement the security-enforcing TOE security functional requirements.

337 To provide context for the description of the TSF, the high-level design describes the entire TOE at a high level. From this description the reader will know which subsystems are part of the TSF and those that are not. The remainder of the high-level design document then describes the TSF in more detail.

338 The high-level design refines the functional specification into subsystem descriptions. The functional specification provides a description of *what* the TSF does at its interface; the high-level design provides more insight into the TSF by describing *how* the TSF works in order to perform the functions specified at the TSFI. For each subsystem of the TSF, the high-level design identifies the TSFI implemented in the subsystem, describes the purpose of the subsystem and how the implementation of the TSFI (or portions of the TSFI) is designed. The interrelationships of subsystems are also defined in the high-level design. These interrelationships will be represented as data flows, control flows, etc. among the subsystems. It

[23] *This may even be impossible, if the error message is for a condition that the programmer does not expect to occur, but is inserted as part of "defensive programming."*

should be noted that this description is at a high level; low-level implementation detail is not necessary at this level of abstraction.

Application notes

339 The developer is expected to describe the design of the TSF in terms of subsystems. The term "subsystem" is used here to express the idea of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have "subsystems", the developer is expected to represent a similar level of decomposition. For example, a design may be similarly decomposed using "layers", "domains", or "servers".

340 A TSF subsystem is a subsystem that provides mechanisms for enforcing an element of the TSP, or directly supports a subsystem that is responsible for enforcing the TSP. If a subsystem provides a security enforcing interface, then the subsystem is security enforcing. If a subsystem does not provide any security enforcing TSFIs, its mechanisms still must preserve the security of the TSF; such subsystems are termed security supporting.

341 As was the case with ADV_FSP_EXP, the set of SFRs that determine the TSP for the purposes of this component do not include FPT_SEP and FPT_RVM. Those two architectural functional requirements require a different type of analysis than that needed for all other SFRs. A security-enforcing subsystem is one that is designed to implement an SFR other than FPT_SEP and FPT_RVM; the design information and justification for the FPT_SEP and FPT_RVM requirements is given as a result of the ADV_ARC_EXP component.

342 The ADV_HLD_EXP component requires that the developer must identify all subsystems of the TSF (not just the security-enforcing ones). In general, the component requires that the security-enforcing aspects of the subsystems be described in more detail than the security-supporting aspects. The descriptions for the security-enforcing aspects should provide the reader with enough information to determine *how* the implementation of the SFRs is designed, while the description for the security-supporting aspects should provide the reader enough assurance to determine that the subsystems or portions of subsystems that are security supporting have been correctly classified.

343 The ADV_HLD_EXP.1.2E element for this component defines a requirement that the evaluator determine that the high-level design is an accurate and complete instantiation of the user-visible TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the high-level design. The requirement for completeness is intended to be relative to the level of abstraction of the high-level design.

E.4 Rationale for ADV_INT_EXP.1

344 This explicit component was created to levy different modularity metrics on the TSF modules that enforce the identified SFPs[24] versus those that do not[25] (SFP-enforcing modules versus non-SFP-enforcing modules).

[24] These "*SFP enforcing modules*" are all the modules that implement the mechanisms that enforce the discretionary access control and the cryptographic policies.

The intent is that all of the modules that play an active role in enforcing the discretionary access control and cryptographic policies are identified as SFP-enforcing. The remaining modules in the TSF are non-SFP-enforcing modules.

Objectives

346 This component addresses the internal structure of the software TSF. The SFP-enforcing modules require adherence to stricter coupling and cohesion metrics than do the non-SFP-enforcing modules due to their key role in policy enforcement. While the non-SFP-enforcing modules also play a role in TOE security, their role is not as critical as the SFP-enforcing modules, therefore, the degree of coupling and cohesion required of these modules is not as restrictive. It is expected that all of the TSF modules are designed using good software engineering practice, whether they are developed by the developer or incorporated as a third party implementation into the TSF.

347 Requirements are presented for modular decomposition of TOE functionality into SFP-enforcing and non-SFP-enforcing functionality. These requirements, when applied to the internal structure of the TSF, should result in improvements in the complete and correct implementation of TOE security functions; in the understanding of the TOE by both the developer and the evaluator; and in providing the basis for designing and evaluating test suites. Further, improving understandability of the TSF should assist the developer in simplifying its maintainability.

348 Modular design aids in achieving understandability by clarifying the dependencies and interactions a module has on other modules (*coupling*), by including in a module only tasks that are strongly related to each other (*cohesion*), and by elucidating the design of a module through internal structuring and reduced complexity. The use of modular design reduces the interdependence between elements of the TSF and thus reduces the risk that a change or error in one module will have effects throughout the TOE. Its use enhances clarity of design and provides for increased assurance that unexpected effects do not occur. An additional desirable property of modular decomposition is reduction in the amount of unneeded code.

349 The incorporation of modular decomposition into the design and implementation process must be accompanied by sound software engineering considerations. A practical, useful software system will usually entail some undesirable coupling among modules, some modules that include loosely-related functions, and some complexity in a module's design. These deviations from the ideals of modular decomposition are often necessary to achieve some goal or constraint, be it related to performance, compatibility, future planned functionality, or other factors, and may be acceptable, based on the developer's justification for them. In applying the requirements of this class, due consideration must be given to sound software engineering principles; however, the overall objective of achieving understandability must be achieved.

350 A key component to reducing complexity is the use of coding standards. Coding standards are used as a reference to ensure programmers generate code that can be easily understood by individuals (e.g., code maintainers, code reviewers, evaluators) that are not intimately familiar with the nuances of the functions performed by the code. Coding standards aid understandability

[25] These "*non-SFP enforcing modules"* are all other modules that either enforce other policies or support the TSP (e.g., provide infrastructure support, such as scheduling, reading binary data from the disk).

and maintainability by ensuring that meaningful names are given to variables and data structures, the code has a structure that is similar to code developed by other programmers, bad coding practices (e.g., leaving a loop to another section of code and returning) are avoided, the use of pointers to variables/data structures is straightforward, and the code is suitably commented (inline and/or by a preamble). The use of coding standards helps to eliminate errors in code development and assists the development team in performing code walk-throughs. Some aspects of coding standards are specific to a given program language. It is expected that the coding standards are appropriately followed. The requirements in this component allow for exceptions to the adherence of coding standards that may be necessary for reasons of performance, or some other factors, but these deviations must be justified (on a per module basis) as to why they are necessary. Any justification provided must address why the deviation does not unduly introduce complexity into the module, since ultimately, the goal of adhering to coding standards is to improve clarity.

351 Design complexity minimization is a key characteristic of a reference validation mechanism[26], the purpose of which is to produce a TSF that is easily understood and can be completely analyzed.

Application notes

352 Several of the elements within this component refer to the architectural description. The architectural description is at a similar level of abstraction as the low-level design, in that it is concerned with the modules of the TSF. Whereas the low-level design describes the design of the modules of the TSF, the purpose of the architectural description is to provide evidence of modular decomposition of the TSF. Both the low-level design and the implementation representation are required to be in compliance with the architectural description, to provide assurance that these TSF representations possess the required modular decomposition.

353 The requirements in this component refer to SFP-enforcing and non-SFP-enforcing portions of the TSF. The non-SFP-enforcing portions of the TSF consist of the TSP-supporting modules and TSP-enforcing modules that do not play a role in the enforcement of the SFP(s) identified in ADV_INT_EXP.1.4D (see Figure **E-1)**.

[26] There are other important characteristics of a reference validation mechanism, such as TSF self-protection and TSP non-bypassability; these other characteristics are covered by requirements from other classes.

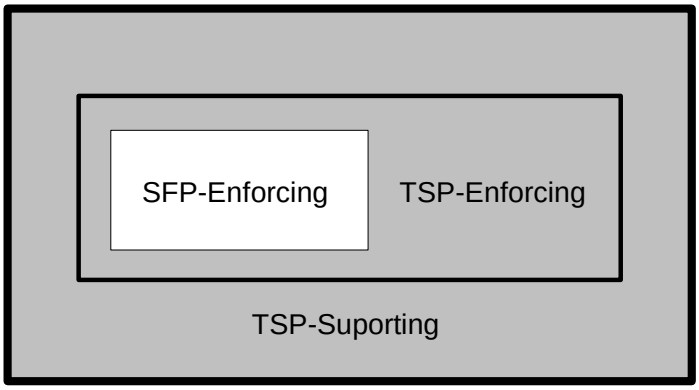

Figure E-1 SFP-enforcing may only be a subset of TSP-enforcing functions.

354 The developer is required to identify all TSF modules as either SFP-enforcing or non-SFP-enforcing.[27] The justification for designating a module as non-SFP-enforcing (ADV_INT_EXP.1.3C) is required only for those modules that interact with SFP-enforcing modules and not for all non-SFP-enforcing modules (see Figure **E-2).**

[27] The modules identified in the architectural description are the same as the modules identified in the low-level design.

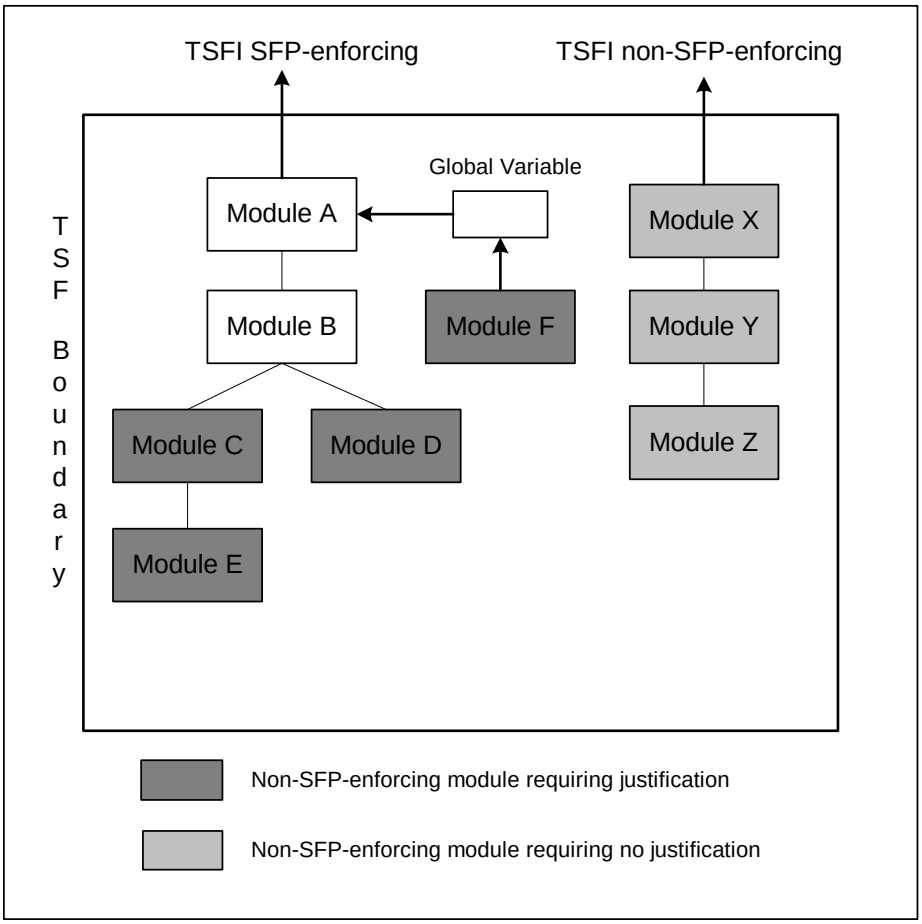

Figure E-2 Example of non-SFP-enforcing modules requiring justification.

Terms, definitions and background

355 The following terms are used in the requirements for software internal structuring. Some of these are derived from the Institute of Electrical and Electronics Engineers *Glossary of software engineering terminology, IEEE Std 610.12-1990.*

module: one or more source code files that cannot be decomposed into smaller compilable units.

modular decomposition: the process of breaking a system into components to facilitate design and development.

*cohesion (*also called *module strength):* the manner and degree to which the tasks performed by a single software module are related to one another; types of cohesion include coincidental, communicational, functional, logical, sequential, and temporal. These types of cohesion are characterized below, listed in the order of decreasing desirability.

functional cohesion: a module with this characteristic performs activities related to a single purpose. A functionally cohesive module transforms a single type of input into a single type of output, such as a *stack manager* or a *queue manager*.

sequential cohesion: a module with this characteristic contains functions each of whose output is input for the following function in the module. An example of a sequentially cohesive module is one that contains the functions to write audit records and to maintain a running count of the accumulated number of audit violations of a specified type.

communicational cohesion: a module with this characteristic contains functions that produce output for, or use output from, other functions within the module. An example of a communicationally cohesive module is an *access check* module that includes mandatory, discretionary, and capability checks.

temporal cohesion: a module with this characteristic contains functions that need to be executed at about the same time. Examples of temporally cohesive modules include *initialization, recovery,* and *shutdown* modules.

logical (or procedural) cohesion: a module with this characteristic performs similar activities on different data structures. A module exhibits logical cohesion if its functions perform related, but different, operations on different inputs.

coincidental cohesion: a module with this characteristic performs unrelated, or loosely related activities.

coupling: the manner and degree of interdependence between software modules; types of coupling include call, common and content coupling. These types of coupling are characterized below, listed in the order of decreasing desirability

call: two modules are call coupled if they communicate strictly through the use of their documented function calls; examples of call coupling are data, stamp, and control, which are defined below.

- *data:* two modules are data coupled if they communicate strictly through the use of call parameters that represent single data items.
- *stamp:* two modules are stamp coupled if they communicate through the use of call parameters that comprise multiple fields or that have meaningful internal structures.
- *control:* two modules are control coupled if one passes information that is intended to influence the internal logic of the other.

common: two modules are common coupled if they share a common data area or a common system resource. Global variables indicate that modules using those global variables are common coupled.[28]

Common coupling through global variables is generally allowed, but only

[28] *It can be argued that modules sharing definitions, such as data structure definitions, are common coupled. However, for the purposes of this analysis, shared definitions are considered acceptable, but are subject to the cohesion analysis.*

to a limited degree. For example, variables that are placed into a global area, but are used by only a single module, are inappropriately placed, and should be removed. Other factors that need to be considered in assessing the suitability of global variables are:

- The number of modules that modify a global variable: In general, only a single module should be allocated the responsibility for controlling the contents of a global variable, but there may be situations in which a second module may share that responsibility; in such a case, sufficient justification must be provided. It is unacceptable for this responsibility to be shared by more than two modules. (In making this assessment, care should be given to determining the module actually responsible for the contents of the variable; for example, if a single routine is used to modify the variable, but that routine simply performs the modification requested by its caller, it is the calling module that is responsible, and there may be more than one such module). Further, as part of the complexity determination, if two modules are responsible for the contents of a global variable, there should be clear indications of how the modifications are coordinated between them.
- The number of modules that reference a global variable: Although there is generally no limit on the number of modules that reference a global variable, cases in which many modules make such a reference should be examined for validity and necessity.

content: two modules are content coupled if one can make direct reference to the internals of the other (e.g. modifying code of, or referencing labels internal to, the other module). The result is that some or all of the content of one module are logically included in the other. Content coupling can be thought of as using unadvertised module interfaces; this is in contrast to call coupling, which uses only advertised module interfaces.

call tree: a diagram that identifies the modules in a system and shows which modules call one another. All the modules named in a call tree that originates with (i.e., is rooted by) a specific module are the modules that directly or indirectly implement the functions of the originating module.

software engineering: the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. As with engineering practices in general, some amount of judgment must be used in applying engineering principles. Many factors affect choices, not just the application of measures of modular decomposition, layering, and minimization. For example, a developer may design a system with future applications in mind that will not be implemented initially. The developer may choose to include some logic to handle these future applications without fully implementing them; further, the developer may include some calls to as-yet unimplemented modules, leaving *call stubs*. The developer's justification for such deviations from well-structured programs will have to be assessed using judgment, as well as the application of good software engineering discipline.

complexity: this is a measure of how difficult software is to understand, and thus to analyze, test, and maintain. Reducing complexity is the ultimate goal for using modular decomposition, layering and minimization. Controlling coupling and cohesion contributes significantly to this goal.

356 A good deal of effort in the software engineering field has been expended in attempting to develop metrics to measure the complexity of source code. Most of these metrics use easily computed properties of the source code, such as the number of operators and operands, the complexity of the control flow graph (*cyclomatic complexity*), the number of lines of source code, the ratio of comments to executable code, and similar measures. Coding standards have been found to be a useful tool in reducing complexity and in generating code that is more readily understood.

357 While this component calls for the evaluator to perform a *complexity analysis*, it is expected that the developer will provide support for the claims that the modules are not overly complex (ADV_INT_EXP.1.3D, ADV_INT_EXP.1.6D, ADV_INT_EXP.1.9C). This support could include the developer's programming standards, and an indication that all modules meet the standard (or that there are some exceptions that are justified by software engineering arguments). It could include the results of tools used to measure some of the properties of the source code. Or it could include other support that the developer finds appropriate.

358 While this component calls for the evaluator to perform a *cohesion analysis*, it is expected that the developer will provide support for the claims that the modules exhibit acceptable types of cohesion (ADV_INT_EXP.1.4D, ADV_INT_EXP.1.8C, ADV_INT_EXP.1.9C).

E.5 Rationale for ADV_LLD_EXP.1

Objectives

359 The low-level design of a TOE provides a description of the internal workings of the TSF in terms of modules, global data, and their interrelationships. The low-level design is a description of *how* the TSF is implemented to perform its functions, rather than *what* the TSF provides as is specified in the FSP. The low-level design is closely tied to the actual implementation of the TSF, unlike the high-level design, which could be implementation-independent. The primary goal of the low-level design is an aid in understanding the implementation of the TSF, both by reviewing the text of the low-level design as well as a guide when examining the implementation representation (source code).

Application notes

360 A module is generally a relatively small architectural unit that exhibits properties discussed in ADV_INT_EXP. A "module" in terms in of the ADV_LLD_EXP requirement refers to the same entity as a "module" for the ADV_INT_EXP requirement.

361 A security-enforcing module is a module that directly implements a security policy of the TOE. If a module of the TSF is not security enforcing, its implementation still must preserve the security of the TSF; such modules are termed security supporting.

362 A description of a security-enforcing module in the low-level design should be of sufficient detail so that one could create an implementation of the module from the low-level design, and that implementation would

- be identical to the actual TSF implementation in terms of the interfaces presented and used by the module, and
- be algorithmically identical[29] to the implementation of the module.

363 Security-supporting modules do not need to be described in the same amount of detail, but they should be identified and enough information should be supplied so that 1) the evaluation team can determine that such modules are correctly classified as security supporting (vs. security enforcing), and 2) the evaluation team has the information necessary to complete the analysis required by ADV_INT_EXP.1.

364 In the low-level design, security-enforcing modules are described in terms of the interfaces they present to other modules; the interfaces they use (call interfaces) from other modules; return values; global data they access; their purpose; and an algorithmic description of how they provide that function. Security supporting modules are described only in terms of the interfaces they present and their purpose.

365 The interfaces presented by a module are those interfaces used by other modules to invoke the functionality provided. Interfaces are described in terms of their parameters, and any values that are returned from the interface. In addition to a list of parameters, the descriptions of these parameters are also given. If a parameter were expected to take on a set of values (e.g., a “flag” parameter), the complete set of values the parameter could take on that would have an effect on module processing would be specified. Likewise, parameters representing data structures are described such that each field of the data structure is identified and described. Note that different programming languages may have additional “interfaces” that would be non-obvious; an example would be operator/function overloading in C++. This “implicit interface” in the class description would also be described as part of the low-level design. Note that although a module could present only one interface, it is more common that a module presents a small set of related interfaces.

366 By contrast, interfaces used by a module must be identified such that it can be determined the unique interface that is being invoked by the module being described. It must also be clear from the low-level design the reason the invoking module is being called (e.g., what is the expected result from the call).

367 If the implementation makes use of global data, the low-level design must describe the global data, and in the algorithmic descriptions of the modules indicate how the specific global data are used by the module. Global data are identified and described much like parameters of an interface.

[29] Algorithmically identical means that the processing performed by all implementations of the module will be equivalent, but specific coding choices and data structures may differ.

368 The purpose of a module is a description indicating what function the module provides. The level of detail should be such that the reader can understand what the module's function is in the architecture, and to determine whether or not it is a security-enforcing module.

369 As discussed previously, the algorithmic description of the module should describe in an algorithmic fashion the implementation of the module. This can be done in pseudo-code, through flow charts, or informal text. It discusses how the parameters to the interface, global data, and called functions are used to accomplish the result. It notes changes to global data, system state, and return values produced by the module. It is at the level of detail that an implementation could be derived that would be very similar to the actual implementation of the system. It does not need to describe actual implementation artifacts (*do* loops vs *for* loops, linked lists vs arrays) if such artifacts are algorithmically identical.

370 It should be noted that source code does not meet the low-level design requirements. Although the low-level design describes the implementation, it *is not* the implementation. Further, the comments surrounding the source code are not sufficient low-level design if delivered interspersed in the source code. The low-level design must stand on its own, and not depend on source code to provide details that must be provided in the low level design (whether intentionally or unintentionally). However, if the comments were extracted by some automated or manual process to produce the low-level design (independent of the source code statements), they could be found to be acceptable if they met all of the appropriate requirements.

371 The ADV_LLD_EXP.1.2E element in this component defines a requirement that the evaluator determine that the low-level design is an accurate and complete instantiation of the user-visible TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the low-level design, in addition to the pairwise correspondences required by the ADV_RCR family. Although the evaluator may use the evidence provided in ADV_RCR as an input to making this determination, ADV_RCR cannot be the basis for a positive finding in this area. The requirement for completeness is intended to be relative to the level of abstraction of the low-level design. Note that for this element, FPT_SEP and FPT_RVM are not explicitly analyzed; the analysis for those requirements is done as part of the activity for the ADV_ARC_EXP component.