U.S. Government

Traffic-Filter Firewall

Protection Profile

For

Medium Robustness Environments

Information
Assurance
Directorate

Version 1.0

February 18, 2005

This Profile supersedes the U.S. Department of Defense Traffic-Filter Firewall Protection Profile for Medium Robustness Environments, Version 1.4, dated May 1, 2000.

Protection Profile Title:

U.S. Government Traffic-Filter Firewall Protection Profile for Medium Robustness Environments.

Criteria Version:

This Protection Profile (PP) was developed using Version 2.1 of the Common Criteria (CC) [1] and applying the NIAP interpretations that have been approved by TTAP/CCEVS Management as of July 10, 2002.

Table of Contents

U.S. GOVERNMENT ..... **I**

TRAFFIC-FILTER FIREWALL ..... **I**

1.0 INTRODUCTION **1**

1.1 PROTECTION PROFILE IDENTIFICATION 1
1.2 PROTECTION PROFILE OVERVIEW 1
1.3 CONVENTIONS 2
1.4 RELATED PROTECTION PROFILES 3
1.5 PROTECTION PROFILE ORGANIZATION 3

2.0 TOE DESCRIPTION **5**

2.1 IDENTIFICATION AND AUTHENTICATION 5
2.2 ADMINISTRATION 5
2.3 INFORMATION FLOW CONTROL 5
2.4 TRUSTED CHANNEL/PATH 6
2.5 ENCRYPTION 6
2.6 AUDIT 7

3.0 TOE SECURITY ENVIRONMENT **8**

3.1 VALUE OF RESOURCES 8
3.2 AUTHORIZATION OF ENTITIES 8
3.3 SELECTION OF APPROPRIATE ROBUSTNESS LEVEL 9
3.4 ASSUMPTIONS 12
3.5 THREATS 12
3.5.1 THREATS ADDRESSED BY THE TOE 14
3.6 ORGANIZATIONAL SECURITY POLICIES 16

4.0 SECURITY OBJECTIVES **17**

4.1 TOE SECURITY OBJECTIVES 17
4.2 SECURITY OBJECTIVES FOR THE OPERATING ENVIRONMENT 19

5 IT SECURITY REQUIREMENTS **21**

5.1 TOE FUNCTIONAL SECURITY REQUIREMENTS 21
5.1.1 SECURITY AUDIT (FAU) 24
5.1.2 CRYPTOGRAPHIC SUPPORT (FCS) 37
5.1.3 USER DATA PROTECTION (FDP) 52
5.1.4 IDENTIFICATION AND AUTHENTICATION (FIA) 61
5.1.5 SECURITY MANAGEMENT (FMT) 64
5.1.6 PROTECTION OF THE TOE SECURITY FUNCTIONS (FPT) 70
5.1.7 RESOURCE ALLOCATION (FRU_RSA) 73
5.1.8 TOE ACCESS (FTA) 74
5.1.9 TRUSTED PATH/CHANNELS (FTP) 75
5.2 SECURITY REQUIREMENTS FOR THE IT ENVIRONMENT 77

5.3 TOE SECURITY ASSURANCE REQUIREMENTS 80
6.0 RATIONALE 100
6.1 RATIONALE FOR TOE SECURITY OBJECTIVES 100
6.2 RATIONALE FOR THE SECURITY OBJECTIVES AND SECURITY FUNCTIONAL REQUIREMENTS FOR THE ENVIRONMENT 113
6.3 RATIONALE FOR TOE SECURITY REQUIREMENTS 113
6.4 RATIONALE FOR ASSURANCE REQUIREMENTS 130
6.5 RATIONALE FOR NOT SATISFYING ALL DEPENDENCIES 131
6.6 RATIONALE FOR STRENGTH OF FUNCTION CLAIM 133
6.7 RATIONALE FOR EXPLICIT REQUIREMENTS 133
7.0 ADV EXPLICIT ASSURANCE BACKGROUND INFORMATION 136
7.1 ADV_INT_EXP 136
7.2 ADV_FSP_EXP.1 144
7.3 ADV_HLD_EXP.1 151
7.4 ADV_LLD_EXP.1 153
7.5 ADV_ARC_EXP.1 156
8.0 REFERENCES 158
9.0 TERMINOLOGY 159
10.0 ACRONYMS 164
11.0 REFINEMENTS 166

List of Tables

TABLE 1 - SECURITY FUNCTIONAL REQUIREMENTS 23
TABLE 2 - EXPLICIT SECURITY FUNCTIONAL REQUIREMENTS 24
TABLE 3 – AUDITABLE EVENTS 34
TABLE 4 – ASSURANCE REQUIREMENTS 81
TABLE 5 - SECURITY OBJECTIVES TO THREATS AND POLICIES MAPPINGS 112
TABLE 6 - RATIONALE FOR TOE SECURITY REQUIREMENTS 130
TABLE 7 - UNSUPPORTED DEPENDENCY RATIONALE 132
TABLE 8 - RATIONALE FOR EXPLICIT REQUIREMENTS 135

1.0 INTRODUCTION

This Traffic-Filter Firewall Protection Profile (PP) for Medium Robustness Environments was generated under the Network Boundary Information Assurance Technologies and Solutions Support (NBIAT&S) Program, sponsored by the National Security Agency (NSA). This Protection Profile is intended to be used as follows:

- For product vendors and security product evaluators, this PP defines the requirements that must be addressed by specific products as documented in vendor Security Targets (STs).

- For system integrators, this PP is useful in identifying areas that need to be addressed to provide secure system solutions. By matching the PP with available STs, security gaps may be identified and products or procedures may be configured to bridge these gaps.

1.1 Protection Profile Identification

Title: U. S. Government Traffic-Filter Firewall Protection Profile (PP) for Medium Robustness Environments

Sponsor: National Security Agency (NSA)

CC Version: Common Criteria (CC) Version 2.1, and applicable interpretations.

Registration: <to be provided upon registration>

Protection Profile Version: Version 1.0, dated February 18, 2005

Keywords: Protection Profile, Boundary Gateway, encryption, decryption, information flow control, firewall, Medium Robustness Environments

1.2 Protection Profile Overview

This PP specifies the minimum-security requirements for network boundary devices that provide controlled connectivity between two or more network environments (hereafter referred to as the Target of Evaluation (TOE)) used by the Department of Defense (DoD) in Medium Robustness Environments. The TOE may be a dedicated device such as a firewall, or an enhancement to some other network device such as a router. The target robustness level of "medium" is specified in the *Guidance and Policy for the Department of Defense Global Information Grid Information Assurance (GIG)* [2] and is further discussed in Section 3.0 of this PP.

The TOE supports user identification and authentication (I&A) where "user" is defined to be a human user acting in a role (i.e., Security Administrator, Cryptographic Administrator, and Audit Administrator) or an authorized IT entity. The TOE provides the capability to pass and block information flows based on a set of rules defined by the Security Administrator. The TOE supports encryption for remote administration and authorized IT entities (e.g., certificate server, NTP server), and generates audit data of security relevant events.

The assurance requirements were originally based upon Evaluated Assurance Level (EAL) 4. In order to gain the necessary level of assurance for medium robustness environments explicit requirements have been created for some families in the ADV class both to remove ambiguity in the existing ADV requirements as well as to provide greater assurance than that associated with EAL4. The assurance requirements are presented in Section 5.3.

This PP defines:

- assumptions about the security aspects of the environment in which the TOE will be used;
- threats that are to be addressed by the TOE;
- security objectives of the TOE and its environment;
- functional and assurance requirements to meet those security objectives; and
- rationale demonstrating how the requirements meet the security objectives, and how the security objectives address the threats.

1.3 Conventions

The notation, formatting, and conventions used in this PP are largely consistent with those used in version 2.1 of the Common Criteria (CC). Selected presentation choices are discussed here to aid the PP user.

The CC allows several operations to be performed on functional requirements; *refinement*, *selection*, *assignment*, and *iteration* are defined in paragraph 2.1.4 of Part 2 of the CC. Each of these operations is used in this PP.

The **refinement** operation is used to add detail to a requirement, and thus further restricts a requirement. Refinement of security requirements is denoted by **bold text**.

The **selection** operation is used to select one or more options provided by the CC in stating a requirement. Selections that have been made by the PP authors are denoted by *italicized text*, selections to be filled in by the ST author appear in square brackets with an indication that a selection is to be made, [selection:], and are not italicized.

The **assignment** operation is used to assign a specific value to an unspecified parameter, such as the length of a password. Assignments that have been made by the PP authors are denoted by showing the value in square brackets, [Assignment_value], assignments to be filled in by the ST author appear in square brackets with an indication that an assignment is to be made [assignment:].

The **iteration** operation is used when a component is repeated with varying operations. Iteration is denoted by showing the iteration number in parenthesis following the component identifier, (iteration_number).

As this PP was sponsored, in part by NSA, National Information Assurance Partnership (NIAP) interpretations are used and are presented with the NIAP interpretation number as part of the requirement identifier (e.g., **FAU_GEN.1-NIAP-0410** for Audit data generation).

The CC paradigm also allows protection profile and security target authors to create their own requirements. Such requirements are termed 'explicit requirements' and are permitted if the CC does not offer suitable requirements to meet the authors' needs. Explicit requirements must be identified and are required to use the CC class/family/component model in articulating the requirements. In this PP, explicit requirements will be indicated with the "EXP" following the component name.

Application Notes are provided to help the developer, either to clarify the intent of a requirement, identify implementation choices, or to define "pass-fail" criteria for a requirement. For those components where Application Notes are appropriate, the Application Notes will follow the requirement component.

1.4 Related Protection Profiles

The following Protection Profiles are related to this PP:

- U.S. Department of Defense Traffic-Filter Firewall Protection Profile for Medium Robustness Environments [3].

- U.S. Department of Defense Application-Level Firewall Protection Profile for Medium Robustness Environments Draft [4].

These two firewall PP's for Medium Robustness Environments both provide background source material for the development of this PP.

1.5 Protection Profile Organization

Section 1, Protection Profile Introduction, provides the document management and overview information necessary to identify the PP along with references to other related PP's.

Section 2, Target of Evaluation (TOE) Description, defines the TOE and establishes the context of the TOE by referencing generalized security requirements.

Section 3, TOE Security Environment (TSE), describes the expected environment in which the TOE is to be used. This section defines the set of threats that are relevant to the secure operation of the TOE, organizational security policies with which the TOE must comply, and secure usage assumptions applicable to this analysis.

Section 4, Security Objectives, defines the set of security objectives to be satisfied by the TOE and by the TOE operating environment.

Section 5, IT Security Requirements, specifies the security functional and assurance requirements that must be satisfied by the TOE and the IT environment.

Section 6, Rationale, provides rationale to demonstrate that the security objectives satisfy the threats and policies. This section also explains how the set of requirements are complete relative to the security objectives and presents a set of arguments that address dependency analysis and Strength of Function (SOF) and use of the explicit requirement.

Section 7, ADV Explicit Assurance Requirement Background Information, provides objectives and application notes for the explicit ADV requirements contained in this PP.

Section 8, References, provides background material for further investigation by users of the PP.

Section 9, Terminology, provides a listing of definitions of terms.

Section 10, Acronyms, provides a listing of acronyms used throughout the document.

Section 11, Refinements, identifies the refinements that were made to CC requirements where text is deleted from a requirement.

2.0 TOE DESCRIPTION

TOEs claiming conformance to this Protection Profile (PP) are network boundary devices that provide controlled connectivity between two or more network environments. The TOE may be a dedicated device such as a firewall, or an enhancement to some other network device such as a router. While this document does not dictate vendor implementations of the functional and assurance requirements defined in Section 5, it does require that all hardware and software components necessary to construct a complete TOE are included in any Security Targets (ST) claiming conformance to this PP. The TOE functional requirements can be categorized as follows: Identification and Authentication, Administration, Information Flow Control, Trusted Channel/Path, Encryption, and Audit.

2.1 Identification and Authentication

The TOE may require multiple Identification and Authentication (I&A) mechanisms for access to services residing on the TOE. The type of authentication mechanism required depends on the origin of the source (i.e., remote user, TOE console) requesting the service. A local authentication mechanism is required for administrative access to the TOE through the console. Other authentication methods may be required for remote administration. A UNIX style user ID and password mechanism is an example of a local authentication mechanism. An encrypted channel must be established between the TOE and authorized users to protect the transfer of authentication data.

2.2 Administration

"Administrators" refers to the roles assigned to the individuals responsible for the installation, configuration, and maintenance of the TOE. The TOE requires three separate administrative roles: Cryptographic Administrator, Audit Administrator and Security Administrator. The Cryptographic Administrator is responsible for the configuration and maintenance of cryptographic elements related to the establishment of secure connections to and from the TOE. The Audit Administrator is responsible for the regular review of the TOE's audit data. The Security Administrator is responsible for all other administrative tasks (e.g., creating the TOE security policy) not addressed by the other two administrative roles. It is important to note that while this PP requires the three administrative roles outlined above, it provides the ST author the option of including additional administrative roles as well.

2.3 Information Flow Control

Section 5.1 "TOE Functional Security Requirements" defines the minimum set of configurable security attributes required to permit or deny information flows to or through the TOE. The set of security attributes will include items such as source and destination identification, service identifiers, and user authentication. The TOE Security Administrator configures the security attributes to construct one or more access control rules as part of a security policy on the TOE. While vendor implementation of the rules and security policy is not dictated, a typical implementation may consist of one or more "rulesets" that are subsequently applied to one or more TOE interfaces. Packets arriving at the TOE interface are compared to the security attributes in the "rulesets". When the packet attributes "match" the rules security attributes, that

packet or connection is approved. In addition to restricting access via the rules, the TOE must generate and maintain "state" information for all approved connections mediated by the TOE. The TOE utilizes the "state" information to monitor the status of an approved connection and validate incoming packets purporting to be part of an approved connection. The FDP_IFF.1.3-NIAP-0417 requirement defines the minimum sets of "state" attributes required by the TOE. Additional TOE requirements such as an inactivity timer and controls on half-open connections are included to assist the Security Administrator with managing the resources utilized by maintaining "state" information. The TOE is required to perform a complete reassembly of all packet fragments prior to making an access control decision on the packet.

As mentioned at the beginning of the previous paragraph, the PP defines a minimal set of information flows to TOE. The same security attributes discussed above apply to controlling access to services residing on the TOE. This PP includes Internet Control Message Protocol (ICMP) as a required unauthenticated information flow to the TOE. The TOE must provide the Security Administrator with the capability of enabling or disabling ICMP data to or from the TOE. When ICMP is enabled, the security attributes defined in the FDP_IFF.1.3-NIAP-0417(3) requirement, to include control of the ICMP message types must be available to the Security Administrator. The ST author may include additional unauthenticated information flows to the TOE but they must meet the requirements in FDP_IFF.1.3-NIAP-0417(2). Remote administration is a required information flow to the TOE, authentication/certificate servers, Network Time Protocol (NTP) servers, as well as any other IT entities are optional.

2.4 Trusted Channel/Path

The TOE is required to provide two types of encrypted communications: trusted channel and trusted path. Trusted channel refers to the encrypted connection between the TOE and a non-human external source. An encrypted connection between the TOE and authorized IT entities (e.g., NTP server, certificate authority) is an example of trusted channel encryption. Trusted path refers to the encrypted connection used to authenticate an external human user with the TOE. Remote administrators establishing an encrypted link to authenticate to the TOE are examples of trusted path encryption. The remote administrator's communication must remain encrypted throughout the remote session.

2.5 Encryption

As mentioned in section 2.4 above, the TOE must establish encrypted communications (acting as the initiator or responder) with external authorized IT entities. Section 5.1.2 "Cryptographic Support" defines the minimum set of cryptographic attributes required by the TOE. The TOE's cryptographic module(s) must be FIPS PUB 140-2 validated and must meet, as a minimum, the security requirements of "Security Level 1". The ST author may implement the cryptographic module(s) in hardware, software, or a combination of both. The TOE must generate and distribute symmetric and asymmetric keys. The ST author is provided several implementation selections for key generation and may distribute keys manually, electronically, or both. The TOE must perform data encryption/decryption using the Advanced Encryption Standard (AES) algorithm with a minimum key size of 128 bits. Additional requirements for key destruction, digital signature generation/verification, random number generation and cryptographic hashing are provided in section 5.1.2.

2.6 Audit

Section 5.1.1 “Security Audit (FAU)” describes the TOE’s generation of auditable events, audit records, alarms and audit management. Table 3 in the FAU_GEN.1-NIAP-0410 requirement lists the minimum set of auditable events that must be available to the Security Administrator for configuration on the TOE. Each auditable event must generate an audit record. Table 3 also provides a minimum list of attributes that must be included in each audit record. The ST author may include additional auditable events and audit record attributes. If the ST author includes any additional functional requirements not specified by this PP, they must consider any security relevant events associated with those requirements and include them in the TOE’s list of auditable events and records. In addition to generating auditable events, the TOE must monitor their occurrences and provide a Security Administrator configurable threshold for determining a potential security violation. Once the TOE has detected a potential security violation, an alarm is generated and a message is displayed at the TOE’s local console as well as each active remote administrator console (all administrative roles included). Additionally, the Security Administrator can configure the TOE to generate an audible alarm to indicate a potential security violation. If an administrator console is not active, the TOE stores the message for display when the console becomes active (e.g. when the administrator establishes a remote session to the TOE). The message must contain the potential security violation and all audit records associated with the potential security violation. The message will be displayed at the various consoles until administrator acknowledgement of the message has occurred. As mentioned in the “Administrative” section above, the Audit Administrator’s role is restricted to viewing the contents of the audit records and the deletion of the audit trail. The TOE does provide the Audit Administrator with a sorting and searching capability to improve audit analysis. The Security Administrator configures auditable events, backs-up and deletes audit data, and manages audit data storage. The TOE provides the Security Administrator with a configurable audit trail threshold to track the storage capacity of the audit trail. As soon as the threshold is met, the TOE generates an alarm and displays a message in the same fashion as described above, including the option of the audible alarm. In addition to displaying the message, the Security Administrator may configure the TOE to prevent all auditable events except for those performed by the Security and Audit Administrators or overwrite the oldest audit records in the audit trail.

3.0 TOE SECURITY ENVIRONMENT

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

In general terms, the environment for a TOE can be characterized by the authorization (or lack of authorization) the least trustworthy entity has with respect to the highest value of TOE resources (i.e. the TOE itself and all of the data processed by the TOE).

Note that there are an infinite number of combinations of entity authorization and value of resources; this conceptually "makes sense" because there are an infinite number of potential environments, depending on how the resources are valued by the organization, and the variety of authorizations the organization defines for the associated entities. In Section 3.3, these two environmental factors will be related to the robustness required for selection of an appropriate TOE.

3.1 Value of Resources

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. 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. 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 was being depended upon to protect the high value data, then it must be treated as a high-value-data TOE.

3.2 Authorization of Entities

Authorization that entities (administrators and 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 a router non-administrative entities may have their packets routed by the TOE, but that is the extent of their authorization to the TOE's resources. In the case of an OS, an entity may not be allowed to log on to the TOE at all (that is, they are not valid users listed in the OS's user database)**.**

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 were 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.

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 compromise of the data (that is, compromise of any of the applicable security policies; e.g., confidentiality, integrity, availability). In other words, in this model the greater the extent of an entity's authorization, the more trustworthy (with respect to applicable policies) that entity is.

3.3 Selection of appropriate Robustness level

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.

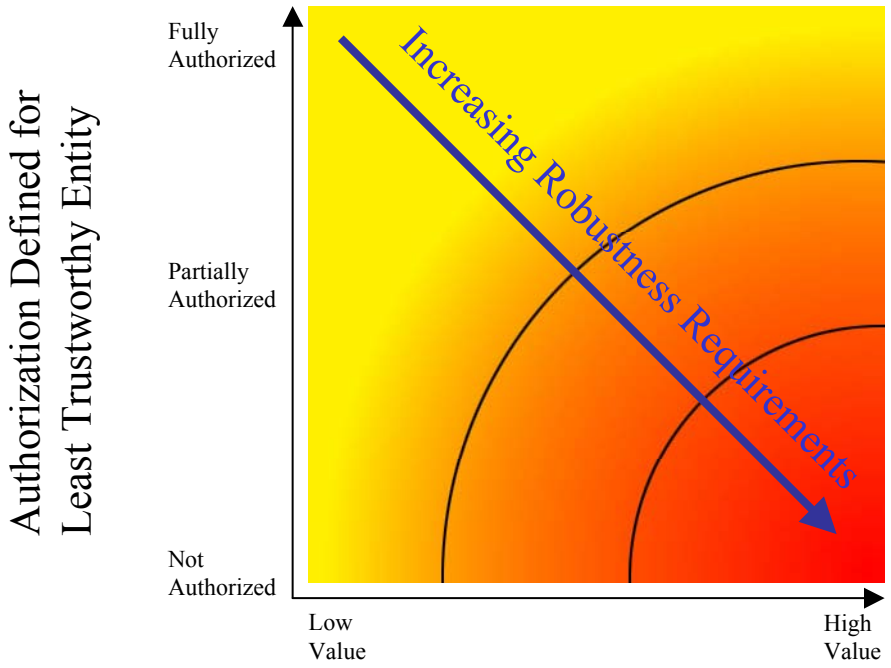
When assessing any environment with respect to Information Assurance the critical point to consider is the likelihood of an attempted security policy compromise, which was characterized in the previous section in terms of entity authorization and resource value. As previously mentioned, robustness is 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.

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:

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.

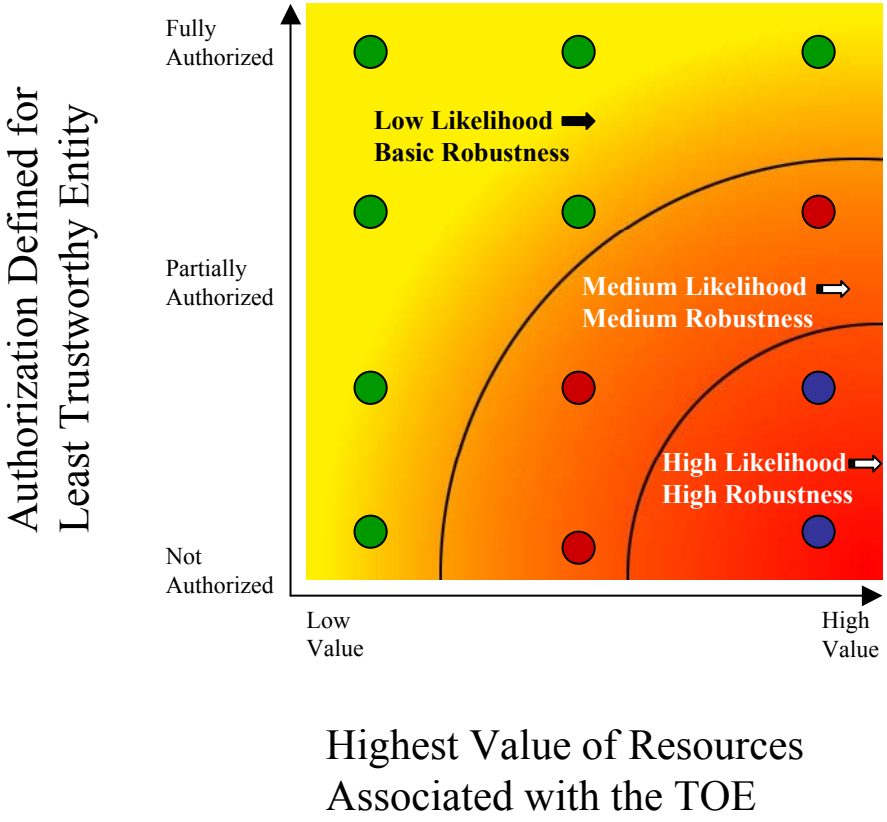
The second case is a TOE that processes high-value (e.g., classified) information. The organization requires that the TOE be stand-alone, and that every user with physical and logical access to the TOE undergo an investigation so that they are authorized to the highest value data on the TOE. Because 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 users and once again selection of a basic robustness TOE would be appropriate.

The preceding examples demonstrated that it is possible for radically different combinations of entity authorization/resource values to result in a similar likelihood of an attempted compromise. As mentioned earlier, the robustness of a system is an indication of the protection being provided to counter compromise attempts. Therefore, a basic robustness system should be sufficient to counter compromise attempts where the likelihood of an attempted compromise is low. The following chart 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.



As depicted in this figure, 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.

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 or 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 set of environments where the likelihood of attempted compromise is roughly



similar. This is graphically depicted in the picture above.

In this second representation of environments and the robustness plane, 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.

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 3.5 of this PP, the targeted threat level for a medium robustness Firewall

TOE is characterized. This information is provided to help organizations insure that the functional requirements specified by this medium robustness PP are appropriate for their intended application of a compliant Firewall.

It is important to note to vendors and end users that any IT entity that is used to protect National Security information, and employs cryptography as a protection mechanism, will require the TOE's key management techniques to be approved by NSA when the TOE is fielded.

The remainder of this section addresses the following:

- Assumptions about the security aspects of a compliant TOE environment;
- Threats to TOE assets or to the TOE environment which must be countered; and
- Organizational security policies that compliant TOEs must enforce.

3.4 Assumptions

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

A.NO_GENERAL_PURPOSE	The Administrator ensures there are no general purpose computing or storage repository capabilities (e.g., compilers, editors, web servers, database servers or user applications) available on the TOE.
A.PHYSICAL	Physical security, commensurate with the value of the TOE and the data it contains, is assumed to be provided by the environment.
A.NO_TOE_BYPASS	Information cannot flow between external and internal networks located in different enclaves without passing through the TOE.

3.5 Threats

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).

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 no authorization to low value data none-the-less has low motivation to compromise the data; thus

a basic robustness TOE should offer sufficient protection. Likewise, the fully authorized user with access to highly valued data similarly has low motivation to attempt to compromise the data, thus again a basic robustness TOE should be sufficient.

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.

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.***

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 (for example, hacking into a bank to obtain money in order to obtain other resources).

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.

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.

The important general points we can 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 are "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 availability of attacks associated with high expertise and/or high availability of resources (for example, via the Internet or "hacker chat rooms") introduces a problem when trying to define the expertise of, or resources available to, a threat agent.

3.5.1 Threats Addressed by the TOE

The following threats are addressed by the TOE and should be read in conjunction with the threat rationale section. There are other threats that the TOE does not address (e.g., malicious developer inserting a backdoor into the TOE) and it is up to a site to determine how these types of threats apply to its environment.

T.ADDRESS_MASQUERADE	A user on one interface may masquerade as a user on another interface to circumvent the TOE policy.
T.ADMIN_ERROR	An administrator may incorrectly install or configure the TOE, or install a corrupted TOE resulting in ineffective security mechanisms.
T.ADMIN_ROGUE	An 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 compromise the cryptographic mechanisms and the data protected by those mechanisms.
T.MASQUERADE	A user may masquerade as an authorized user or an authorized IT entity to gain access to data or TOE resources.

T.FLAWED_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.FLAWED_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 (including in a fielded TOE) may result in incorrect TOE behavior being undiscovered.
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 (captured as it was transmitted during the course of legitimate use).
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 TOE system resources (e.g., connection state tables) via a resource exhaustion denial of service attack.
T.SPOOFING	An entity may mis-represent itself as the TOE to obtain authentication data.
T.MALICIOUS_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 access to services (by sending data through or to the TOE) for which they are not authorized according to the TOE security policy.
T.UNIDENTIFIED_ACTIONS	The administrator may fail to notice potential security violations, thus limiting the

	administrator's ability to identify and take action against a possible security breach.
T.UNKNOWN_STATE	When the TOE is initially started or restarted after a failure, design flaws, or improper configurations may cause the security state of the TOE to be unknown.

3.6 Organizational Security Policies

PP-compliant TOEs must address the organizational security policies described below.

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 system.
P.ACCOUNTABILITY	The authorized users of the TOE shall be held accountable for their actions within the TOE.
P.ADMIN_ACCESS	Administrators shall be able to administer the TOE both locally and remotely through protected communications channels.
P.CRYPTOGRAPHIC_FUNCTIONS	The TOE shall provide cryptographic functions for its own use, including encryption/decryption and digital signature operations.
P.CRYPTOGRAPHY_VALIDATED	Where the TOE requires FIPS-approved security functions, only NIST FIPS validated cryptography (methods and implementations) are acceptable for key management (i.e.; generation, access, distribution, destruction, handling, and storage of keys) and cryptographic services (i.e.; encryption, decryption, signature, hashing, key distribution, and random number generation services).
P.VULNERABILITY_ANALYSIS_TEST	The TOE must undergo appropriate independent vulnerability analysis and penetration testing to demonstrate that the TOE is resistant to an attacker possessing a medium attack potential.

4.0 SECURITY OBJECTIVES

This chapter describes the security objectives for the TOE and the TOE's operating environment. The security objectives are divided between TOE Security Objectives (i.e., security objectives addressed directly by the TOE) and Security Objectives for the Operating Environment (i.e., security objectives addressed by the IT domain or by non-technical or procedural means).

4.1 TOE Security Objectives

This section defines the security objectives that are to be addressed by the TOE.

O.ROBUST_ADMIN_GUIDANCE	The TOE will provide administrators with the necessary information for secure delivery and management.
O.ADMIN_ROLE	The TOE will provide an administrator role to isolate administrative actions.
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 the capability to test the TSF to ensure the correct operation of the TSF in its operational environment.
O.CRYPTOGRAPHIC_ FUNCTIONS	The TOE shall provide cryptographic functions for its own use, including encryption/decryption and digital signature operations.
O.CRYPTOGRAPHY_ VALIDATED	The TOE shall use NIST FIPS 140-2 validated cryptomodules for cryptographic services implementing FIPS-approved security functions and random number generation services used by

	cryptographic functions.
O.DISPLAY_BANNER	The TOE will display an advisory warning regarding use of the TOE.
O.DOCUMENT_KEY_LEAKAGE	The bandwidth of channels that can be used to compromise key material shall be documented.
O.THOROUGH_FUNCTIONAL_ TESTING	The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.
O.MAINT_MODE	The TOE shall provide a mode from which recovery or initial startup procedures can be performed.
O.MANAGE	The TOE will provide all the functions and facilities necessary to support the administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.
O.MEDIATE	The TOE must mediate the flow of information between sets of TOE network interfaces or between a network interface and the TOE itself in accordance with its security policy.
O.REPLAY_DETECTION	The TOE will provide a means to detect and reject the replay of TSF data and security attributes.
O.RESIDUAL_INFORMATION	The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.
O.RESOURCE_SHARING	The TOE shall provide mechanisms that mitigate attempts to exhaust connection-oriented resources provided by the TOE (e.g., entries in a connection state table; Transmission Control Protocol (TCP) connections used by proxies).
O.SELF_PROTECTION	The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure.

O.SOUND_DESIGN	The design of the TOE will be the result of sound design principles and techniques; the design of the TOE, as well as the design principles and techniques, are adequately and accurately documented.
O.SOUND_IMPLEMENTATION	The implementation of the TOE will be an accurate instantiation of its design, and is adequately and accurately documented.
O.TIME_STAMPS	The TOE shall provide reliable time stamps and the capability for the administrator to set the time used for these time stamps.
O.ROBUST_TOE_ACCESS	The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate.
O.TRUSTED_PATH	The TOE will provide a means to ensure administrators are not communicating with some other entity pretending to be the TOE, and that the TOE is communicating with an authorized IT entity and not some other entity pretending to be an authorized IT entity.
O.VULNERABILITY_ANALYSIS_TEST	The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.

4.2 Security Objectives for the Operating Environment

This section defines the security objectives that are to be addressed by the IT domain or by non-technical or procedural means. All of the assumptions stated in Section 3.4 are considered to be security objectives for the environment. There is an additional objective for the environment, OE.CRYPTANALYTIC. The mapping and rationale for the security objectives are described in Section 6.

OE.CRYPTANALYTIC	Cryptographic methods used in the IT environment shall be interoperable with the TOE, should be FIPS 140-2 validated and should be

	resistant to cryptanalytic attacks (i.e., will be of adequate strength to protect unclassified Mission Support, Administrative, or Mission Critical data).
OE.NO_GENERAL_PURPOSE	The Administrator ensures there are no general purpose computing or storage repository capabilities (e.g., compilers, editors, or user applications) available on the TOE.
OE.NO_TOE_BYPASS	Information cannot flow between external and internal networks located in different enclaves without passing through the TOE.
OE.PHYSICAL	Physical security, commensurate with the value of the TOE and the data it contains, is assumed to be provided by the IT environment.

5 IT SECURITY REQUIREMENTS

This section provides functional and assurance requirements that must be satisfied by a Protection Profile-compliant TOE. These requirements consist of components from the CC Part 2 and Part 3, NIAP interpreted requirements, and explicit requirements.

5.1 TOE Functional Security Requirements

The functional security requirements for the TOE consist of the following components derived from Part 2 of the CC, summarized in Table 1 below. Table 2 presents the explicit functional requirements specified in this PP.

Functional Components (from CC Part 2)	
FAU_ARP.1	Security alarms
FAU_SAR.1	Audit review
FAU_SAR.2	Restricted audit review
FAU_SAR.3	Selectable audit review
FAU_STG.3	Action in case of possible audit data loss
FCS_CKM.1	Cryptographic key generation (for AES symmetric keys using RNG)
FCS_CKM.4	Cryptographic key destruction
FDP_IFC.1	Subset information flow control (unauthenticated policy)
FDP_IFC.1(2)	Subset information flow control (unauthenticated TOE services policy)
FDP_RIP.2	Full residual information protection
FIA_ATD.1	User attribute definition
FIA_UAU.1	Timing of authentication
FIA_UID.2	User identification before any action
FIA_USB.1	User-Subject Binding
FMT_MOF.1(1)	Management of security functions behavior (TSF non-cryptographic self-test)
FMT_MOF.1(2)	Management of security functions behavior (cryptographic self-test)

Functional Components (from CC Part 2)	
	test)
FMT_MOF.1(3)	Management of security functions behavior (audit and alarms)
FMT_MOF.1(4)	Management of security functions behavior (audit and alarms)
FMT_MOF.1(5)	Management of security functions behavior (audit and alarms)
FMT_MOF.1(6)	Management of security functions behavior (available TOE-services for unauthenticated users)
FMT_MOF.1(7)	Management of security functions behavior (quota mechanism)
FMT_MSA.1	Management of security attributes
FMT_MTD.1(1)	Management of TSF data (non-cryptographic, non-time TSF data)
FMT_MTD.1(2)	Management of TSF data (cryptographic TSF data)
FMT_MTD.1(3)	Management of TSF data (time TSF data)
FMT_MTD.1(4)	Management of TSF data (information flow policy ruleset)
FMT_MTD.2(1)	Management of limits on TSF data (transport-layer quotas)
FMT_MTD.2(2)	Management of limits on TSF data (controlled connection-oriented quotas)
FMT_REV.1	Revocation
FMT_SMR.2	Restrictions on security roles
FPT_RCV.1	Manual Recovery
FPT_RPL.1	Replay detection
FPT_RVM.1	Non-bypassability of the TSP
FPT_SEP.2	SFP domain separation
FPT_STM.1	Reliable time stamps
FRU_RSA.1(1)	Maximum quotas (transport-layer quotas)
FRU_RSA.1(2)	Maximum quotas (controlled connection-oriented quotas)
FTA_SSL.1	TSF-initiated session locking

Functional Components (from CC Part 2)	
FTA_SSL.2	User-initiated locking
FTA_SSL.3	TSF-initiated termination
FTA_TAB.1	Default TOE access banners
FTA_TSE.1	TOE session establishment
FTP_ITC.1(1)	Inter-TSF trusted channel (Prevention of Disclosure)
FTP_ITC.1(2)	Inter-TSF trusted channel (Detection of Modification)
FTP_TRP.1(1)	Trusted path (Prevention of Disclosure)
FTP_TRP.1(2)	Trusted path (Detection of Modification)

Table 1 - Security Functional Requirements

Explicit Functional Components	
FAU_ARP_ACK_EXP.1	Security alarm acknowledgement
FAU_GEN.1-NIAP-0410	Audit data generation
FAU_GEN.2-NIAP-0410	User identity association
FAU_SAA.1-NIAP-0407	Potential violation analysis
FAU_STG.NIAP-0414-1-NIAP-0429	Site-Configurable Prevention of Audit Loss
FAU_SEL.1-NIAP-0407	Selective audit
FAU_STG.1-NIAP-0423	Protected audit trail storage
FCS_BCM_EXP.1	Baseline cryptographic module
FCS_CKM_SYM_EXP.1	Cryptographic Key Establishment for AES symmetric keys
FCS_CKM_ASYM_EXP.1	Cryptographic Key Entry for Digital Signature/verification private keys
FCS_COP_EXP.2	Cryptographic operation (Encryption/Decryption AES)
FCS_COP_EXP.3	Cryptographic operation (Digital Signature Generation/Verification

Explicit Functional Components	
	Generation/Verification
FCS_COP_EXP.5	Cryptographic operation (Random number generator)
FCS_COP_EXP.6	Cryptographic operation (Cryptographic Hash function)
FDP_IFF.1-NIAP-0417(1)	Simple security attributes (unauthenticated policy)
FDP_IFF.1-NIAP-0417(2)	Simple security attributes (unauthenticated TOE services policy)
FIA_AFL.1-NIAP-0425	Authentication failure handling
FIA_UAU_EXP.2	Specified User authentication before any action
FIA_UAU_EXP.5	Authentication mechanism
FMT_MSA.3-NIAP-0409(1)	Static attribute initialization (ruleset)
FMT_MSA.3-NIAP-0409(2)	Static attribute initialization (services)
FPT_TST_EXP.4	TSF testing (with cryptographic integrity verification)
FPT_TST_EXP.5	Cryptographic self-test

Table 2 - Explicit Security Functional Requirements

5.1.1 Security Audit (FAU)

FAU_ARP.1 Security alarms

FAU_ARP.1.1 – **Refinement:** The TSF shall [immediately display an alarm message, identifying the potential security violation and make accessible the audit record contents associated with the auditable event(s) that generated the alarm, at the:

- local console,
- remote administrator sessions that exist, and;
- remote administrator sessions that are initiated before the alarm has been acknowledged, and;
- at the option of the Security Administrator, generate an audible alarm, and;
- [assignment: other methods]] upon detection of a potential security violation.

Application Note: The TSF provides a message to the local console regardless of

whether an administrator is logged in. The message is displayed at the remote console if an administrator is already logged in, or when an administrator logs in if the alarm message has not been acknowledged. The audit records contents associated with the alarm may or may not be part of the message displayed, however the relevant audit information must be available to administrators. In addition, the TOE provides an audible alarm that can be configured to sound an alarm if desired by the Security Administrator. It is acceptable for the ST author to fill the open assignment with none, if no other methods (e.g., pager, e-mail) are included in the TOE.

Explicit: Security alarm acknowledgement (FAU_ARP_ACK_EXP.1)

FAU_ARP_ACK_EXP.1.1 – The TSF shall display the alarm message identifying the potential security violation and make accessible the audit record contents associated with the auditable event(s) until it has been acknowledged. An audible alarm will sound until acknowledged by an administrator.

FAU_ARP_ACK_EXP.1.2 – The TSF shall display an acknowledgement message identifying a reference to the potential security violation, a notice that it has been acknowledged, the time of the acknowledgement and the user identifier that acknowledged the alarm, at the:

- local console, and
- remote administrator sessions that received the alarm.

Application Note: This explicit requirement is necessary since a CC requirement does not exist to ensure an administrator will be aware of the alarm. The intent is to ensure that if an administrator is logged in and not physically at the console or remote workstation the message will remain displayed until they have acknowledged it. The message will not be scrolled off the screen due to other activity taking place (e.g., the Audit Administrator is running an audit report). If the Security Administrator configures the TOE to generate an audible alarm, the alarm will sound until an administrator acknowledges the alarm. Acknowledging the message and audible alarm could be a single event, or different events.

FAU_ARP_ACK_EXP.1.2 ensures that each administrator that received the alarm message also receives the acknowledgement message, which includes some form of reference to the alarm message, who acknowledged the message and when.

FAU_GEN.1-NIAP-0410 Audit data generation

FAU_GEN.1.1-NIAP-0410 – **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) All auditable events **listed in Table 3**;

c) [selection: [assignment: events at a basic level of audit introduced by the inclusion of additional SFRs], [assignment: events commensurate with a basic level of audit introduced by the inclusion of explicit requirements], no additional events].

Application Note: For the first assignment in the selection, the ST author augments the table (or lists explicitly) the audit events associated with the basic level of audit for any SFRs that the ST author includes that are not included in this PP.

Likewise, for the second assignment the ST author includes audit events that may arise due to the inclusion of any explicit requirements not already in the PP. Because "basic" audit is not defined for such requirements, the ST author will need to determine a set of events that are commensurate with the type of information that is captured at the basic level for similar requirements. It is acceptable for the ST author to choose "no additional events", if the ST author has not included additional requirements, or has included additional requirements that do not have a basic level (or commensurate level) of audit associated with them.

FAU_GEN.1.2-NIAP-0410 - **Refinement**: 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 (if applicable), and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [information specified in column three of Table 3 below].

Application Note: In column 3 of the table below, "if applicable" is used to designate data that should be included in the audit record if it "makes sense" in the context of the event that generates the record. For example, in FDP_IFF, packets may be allowed to flow that do not have a transport layer component (e.g., an ICMP Echo request). For those packets, there is nothing to record with respect to the transport layer abstractions.

Requirement	Auditable Events	Additional Audit Record Contents

Requirement	Auditable Events	Additional Audit Record Contents
FAU_ARP.1	Potential security violation was detected	Identification of what caused the generation of the alarm
FAU_ARP_ACK_EXP.1	None	The identity of the administrator that acknowledged the alarm.
FAU_GEN.1-NIAP-0410	None	
FAU_GEN.2-NIAP-0410	None	
FAU_SAA.1-NIAP-0407	Enabling and disabling of any of the analysis mechanisms	The identity of the Security Administrator performing the function
FAU_SAR.1	Opening the audit trail	The identity of the Administrator performing the function
FAU_SAR.2	Unsuccessful attempts to read information from the audit records	The identity of the administrator attempting the function
FAU_SAR.3	None	
FAU_SEL.1-NIAP-0407	All modifications to the audit configuration that occur while the audit collection functions are operating	The identity of the Security Administrator performing the function
FAU_STG.1-NIAP-0423	None	
FAU_STG.3	Actions taken due to exceeding the audit threshold	The identity of the Security Administrator performing the function
FAU_STG.NIAP-0414-1-NIAP-0429	Actions taken due to the audit storage failure	The identity of the Security Administrator performing the function
FCS_BCM_EXP.1	None	
FCS_CKM.4	None	
FCS_CKM_SYM_EXP.1	Failure of the activity	
FCS_CKM_ASYM_EXP.1	Failure of the activity	

Requirement	Auditable Events	Additional Audit Record Contents
FCS_COP.EXP.2	Failure of cryptographic operation	Type of cryptographic operation Any applicable cryptographic mode(s) of operation, excluding any sensitive information
FCS_COP.EXP.3	Failure of cryptographic operation	Type of cryptographic operation Any applicable cryptographic mode(s) of operation, excluding any sensitive information
FCS_COP.EXP.5	Failure of cryptographic operation	Type of cryptographic operation Any applicable cryptographic mode(s) of operation, excluding any sensitive information
FCS_COP.EXP.6	Failure of cryptographic operation	Type of cryptographic operation Any applicable cryptographic mode(s) of operation, excluding any sensitive information
FDP_IFC.1(1)	None	
FDP_IFC.1(2)	None	

Requirement	Auditable Events	Additional Audit Record Contents
FDP_IFF.1-NIAP-0417(1)	Decisions to permit/deny information flows Failure to reassemble fragmented packets	Presumed identity of source subject Identity of destination subject Transport layer protocol, if applicable Source subject service identifier, if applicable Destination subject service identifier, if applicable Identity of the firewall interface associated on which the TOE received the packet Identity of the rule that allowed or disallowed the packet flow Reason why fragmented packets could not be reassembled (i.e., invalid fragment identifier, invalid offset, invalid fragment data length)

Requirement	Auditable Events	Additional Audit Record Contents
FDP_IFF.1-NIAP-0417(2)	Decisions to permit/deny information flows between a subject and the TOE	Presumed identity of source subject Identity of destination subject Transport layer protocol, if applicable Source subject service identifier, if applicable Destination subject service identifier, if applicable Identity of the firewall interface associated on which the TOE received the packet Identity of the rule that allowed or disallowed the packet flow, if applicable[1]
FDP_RIP.2	None	
FIA_AFL.1-NIAP-0425	The reaching of the threshold for the unsuccessful authentication attempts The actions (e.g. disabling of an account) taken The subsequent, if appropriate, restoration to the normal state (e.g. re-enabling of an account)	Identity of the unsuccessfully authenticated user
FIA_ATD.1	None	
FIA_UAU.1	None	
FIA_UAU_EXP.2	Successful and unsuccessful use of authentication mechanisms	Claimed identity of the user using the authentication mechanism

[1] The TOE may not use a rule in a ruleset to allow/disallow TOE services (e.g., configuration parameter could be used instead) and if this is the case, it is not required that a rule be identified.

Requirement	Auditable Events	Additional Audit Record Contents
FIA_UAU_EXP.5	All use of the local authentication mechanism	Claimed identity of the user attempting to authenticate
FIA_UID.2	All use of the user identification mechanism used for authorized users (that is, those that authenticate to the TOE)	Claimed identity of the user using the identification mechanism
FIA_USB.1	Success and failure of binding of user security attributes to a subject	The identity of the user whose attributes are attempting to be bound
FMT_MOF.1(1)	All modifications in the behavior of the functions in the TSF	The identity of the administrator performing the function
FMT_MOF.1(2)	Enabling or disabling of the key-generation self-tests	The identity of the administrator performing the function
FMT_MOF.1(3)	All modifications in the behavior of the functions in the TSF	The identity of the administrator performing the function
FMT_MSA.1	All manipulation of the security attributes	The identity of the administrator performing the function
FMT_MSA.3-NIAP-0409(1)	None	
FMT_MSA.3-NIAP-0409(2)	None	
FMT_MTD.1(1)	All modifications of the values of TSF data by the administrator	The identity of the administrator performing the function
FMT_MTD.1(2)	All modifications of the values of cryptographic security data by the cryptographic administrator	The identity of the administrator performing the function
FMT_MTD.1(3)	All modifications to the time and date used to form the time stamps by the administrator	The identity of the administrator performing the function
FMT_MTD.1(4)	All modifications to the information flow policy ruleset by the Security Administrator	The identity of the security administrator performing the function

Requirement	Auditable Events	Additional Audit Record Contents
FMT_MTD.2(1)	All modifications of the limits Actions taken when the quota is exceed (include the fact that the quota was exceeded)	The identity of the administrator performing the function
FMT_MTD.2(2)	All modifications of the limits Actions taken when the quota is exceed (include the fact that the quota was exceeded)	The identity of the administrator performing the function
FMT_REV.1	All attempts to revoke security attributes	List of security attributes that were attempted to be revoked The identity of the administrator performing the function
FMT_SMR.2	Modifications to the group of users that are part of a role	User IDs that are associated with the modifications The identity of the administrator performing the function
FPT_RCV.1	The fact that a failure or service discontinuity occurred Resumption of the regular operation	Type of failure or service discontinuity
FPT_RPL.1	Notification that a replay event occurred	Identity of the user that was the subject of the reply attack
FPT_RVM.1	None	
FPT_SEP.2	None	
FPT_STM.1	Changes to the time	
FPT_TST_EXP.4	Execution of this set of TSF self tests	The identity of the administrator performing the test, if initiated by an administrator

Requirement	Auditable Events	Additional Audit Record Contents
FPT_TST_EXP.5	Execution of this set of TSF self tests	The identity of the administrator performing the test, if initiated by an administrator
FRU_RSA.1(1)	None	
FRU_RSA.1(2)	None	
FTA_SSL.1	Locking of an interactive session by the session locking mechanism Any attempts at unlocking of an interactive session	The identity of the user associated with the session being locked or unlocked
FTA_SSL.2	Locking of an interactive session by the session locking mechanism Any attempts at unlocking of an interactive session	The identity of the user associated with the session being locked or unlocked
FTA_SSL.3	The termination of a remote session by the session locking mechanism	The identity of the user associated with the session that was terminated
FTA_TAB.1	None	
FTA_TSE.1	All attempts at establishment of a user session	The identity of the user attempting to establish the session For unsuccessful attempts, the reason for denial of the establishment attempt
FTP_ITC.1(1)	All attempted uses of the trusted channel functions	Identification of the initiator and target of all trusted channels
FTP_ITC.1(2)	All attempted uses of the trusted channel functions	Identification of the initiator and target of all trusted channels
FTP_TRP.1(1)	All attempted uses of the trusted path functions	Identification of the claimed user identity

Requirement	Auditable Events	Additional Audit Record Contents
FTP_TRP.1(2)	All attempted uses of the trusted path functions	Identification of the claimed user identity

Table 3 – Auditable Events

FAU_GEN.2-NIAP-0410 User Identity Association

FAU_GEN.2.1-NIAP-0410 – **Refinement**: 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 association is required because the user is not under TSF control until after a successful identification/authentication. User in this requirement is the userid for authorized users, and a network identifier for unauthenticated network traffic.

FAU_SAA.1-NIAP-0407 Potential violation analysis

FAU_SAA.1.1-NIAP-0407 – The TSF shall be able to apply a set of rules in monitoring events and based upon these rules indicate a potential violation of the TSP.

FAU_SAA.1.2-NIAP-0407 - **Refinement**: The TSF shall enforce the following rules for monitoring events:

a) [Security Administrator specified number of authentication failures;

b) Security Administrator specified number of Information Flow policy violations by an individual presumed source network identifier (e.g., IP address) within an administrator specified time period;

c) Security Administrator specified number of Information Flow policy violations to an individual destination network identifier within an administrator specified time period;

d) Security Administrator specified number of Information Flow policy violations to an individual destination subject service identifier (e.g., TCP port) within an administrator specified time period;

e) Security Administrator specified Information Flow policy rule, or group of rule violations within an administrator specified time period;

f) Any detected replay of TSF data or security attributes;

g) Any failure of the cryptomodule self-tests (FPT_TST_EXP.5);

h) Any failure of the other TSF self-tests (FPT_TST_EXP.4);

i) Security Administrator specified number of encryption failures;

j) Security Administrator specified number of decryption failures;

k) [selection: [assignment: any other rules], "no additional rules"]];

known to indicate a potential security violation.

Application Note: The intent of this requirement is that an alarm is generated (FAU_ARP.1) once the threshold for an event is met. Once the alarm has been generated it is assumed that the "count" for that event is reset to zero. The Security Administrator settable number of authentication failures in (a) is intended to be the same value as specified in FIA_AFL.1.1-NIAP-0425.

FAU_SAR.1 Audit review

FAU_SAR.1.1 – The TSF shall provide [the Administrators] with the capability to read [all audit data] from the audit records.

FAU_SAR.1.2 – **Refinement**: The TSF shall provide the audit records in a manner suitable for the **Administrators** to interpret the information.

Application Note: The role Administrator is intended to mean any user acting in an administrative role.

FAU_SAR.2 Restricted audit review

FAU_SAR.2.1 – **Refinement**: The TSF shall prohibit all users read access to the audit records **in the audit trail**, except **the Administrators**.

FAU_SAR.3 Selectable audit review

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

a) [user identity;

b) source subject identity;

c) destination subject identity;

d) ranges of one or more: dates, times, user identities, subject service identifiers, or transport layer protocol;

e) rule identity;

f) TOE network interfaces; and

g) [selection: [assignment: other criteria], no additional criteria]].

Application Note: Audit data should be capable of being searched and sorted on all criteria specified in a – g, if applicable (i.e., not all criteria will exist in all audit records). Sorting means to arrange the audit records such that they are "grouped" together for administrative review. For example the Audit Administrator may want all the audit records for a specified source subject identity or range of source subject identities (e.g., IP source address or range of IP source addresses) presented together to facilitate their audit review. If no additional criteria are provided by the TOE to perform searches or sorting of audit data, the ST author selects "no additional criteria".

FAU_SEL.1-NIAP-0407 Selective Audit

FAU_SEL.1.1-NIAP-0407 - **Refinement**: The TSF shall **allow only the Security Administrator** to include or exclude auditable events from the set of audited events based on the following attributes:

a) *user identity;*

b) *event type;*

c) [network identifier;

d) subject service identifier;

e) success of auditable security events;

f) failure of auditable security events;

g) rule identity; and

h) [selection: [assignment: list of additional criteria that audit selectivity is based upon], no additional criteria]].

Application Note: "user identity" applies to authenticated users; see application note for FIA_UID.2. "service identifier" is defined in FDP_IFF.1.2-NIAP-0417. "event type" is to be defined by the ST author; the intent is to be able to include or exclude classes of audit events.

FAU_STG.1-NIAP-0423 Protected audit trail storage

FAU_STG.1.1-NIAP-0423 – **Refinement**: The TSF shall **restrict the deletion of** stored audit records in the audit trail **to the Audit Administrator**.

FAU_STG.1.2-NIAP-0423 – **Refinement:** The TSF shall be able to *prevent* modifications to the audit records in the audit trail.

FAU_STG.3 Action in case of possible audit data loss

FAU_STG.3.1 - **Refinement**: The TSF shall [immediately alert the administrators by displaying a message at the local console, and at the remote administrative console when an administrative session exists for each of the defined administrative roles, at the option of the Security Administrator generate an audible alarm, [selection: [assignment: other methods], no other methods] if the audit trail exceeds [a Security Administrator settable percentage of storage capacity].

Application Note: As with FAU_ARP.1, the TSF provides a message to the local console regardless of whether an administrator is logged in. The message is displayed at the remote console if an administrator is already logged in, or when an administrator logs in. This requirement specifies that the message is sent to the first established session for each of the defined roles to ensure someone in the administrator staff is aware of the alert as soon as possible.

FAU_STG.NIAP-0414-1-NIAP-0429 Site-Configurable Prevention of Audit Loss

FAU_STG.NIAP-0414-1.1-NIAP-0429 - **Refinement**: The TSF shall provide the **Security** Administrator the capability to select one or more of the following actions *prevent auditable events, except those taken by the Security Administrator and Audit Administrator, overwrite the oldest stored audit records* and [selection: [assignment: other actions to be taken in case of audit storage failure], no other actions] to be taken if the audit trail is full.

FAU_STG.NIAP-0414-1.2-NIAP-0429 - **Refinement**: The TSF shall **enforce the Security Administrator's selection(s)** if the audit trail is full.

Application Note: The TOE provides the Security Administrator the option of preventing audit data loss by preventing auditable events from occurring. The Security Administrator and Audit Administrator actions under these circumstances are not required to be audited. The TOE also provides the Security Administrator the option of overwriting "old" audit records rather than preventing auditable events, which may protect against a denial-of-service attack.

5.1.2 Cryptographic Support (FCS)

This section specifies the cryptographic support required in the TOE. As previously stated the cryptographic support is required for authentication mechanisms, for trusted path, trusted channel and for integrity mechanisms. The cryptographic requirements are structured to accommodate use of the FIPS 140-2 standard and NIST's Cryptographic Module Validation Program (CMVP) in meeting the requirements, and to accommodate use of multiple cryptographic modules in meeting the required cryptographic functionality.

In general, the required cryptographic functionality is either within the scope of what is currently tested as part of the FIPS 140-2 validation program or the functionality must be evaluated by the CCEVS evaluation process; and the cryptographic functionality is either implemented in a FIPS-validated module or not. As the FIPS 140-2 validation program evolves to handle algorithms and

key sizes not currently covered under FIPS 140-2, it is envisioned that aspects specified in these requirements will eventually be covered by the FIPS program. The following presents the terminology used in the PP to articulate these distinctions

Requirements with FIPS-approved cryptographic functionality:

Cryptographic functionality that is within the scope of what's tested as part of the FIPS 140-2 validation program are *FIPS-approved* cryptographic functions. Defined in FIPS 140-2, an approved cryptographic function is a security function (e.g., cryptographic algorithm, cryptographic key management technique, or authentication technique) that is either:

a) specified in a Federal Information Processing Standard (FIPS),

b) adopted in a FIPS and specified either in an appendix to the FIPS or in a document referenced by the FIPS standard, or

c) specified in the list of Approved security functions.

As specified in P.CRYPTOGRAPHY_VALIDATED, *FIPS-approved* cryptographic functions are required to be implemented in a *FIPs-validated module running in FIPS-approved mode.* FCS_BCM reflects this requirement, and it specifies the required FIPS validation levels for the security functions.

The following requirements specify cryptographic functionality that is currently (August 2003) *FIPS-approved:*

- FCS_CKM.1 (key generation for AES symmetric keys)
- FCS_CKM_ASYM_EXP.1 (key entry for Digital Signature/verification private keys)
- FCS_CKM.4 (key destruction)
- FCS_COP_EXP.2 (encryption/decryption using AES)
- FCS_COP_EXP.3 (digital signature generation/verification)
- FCS_COP_EXP.5 (random number generation)
- FCS_COP_EXP.6 (hashing function)

These requirements specify a *'FIPS-validated cryptomodule'* in the requirement. The requirements also specify the required modes, key sizes, and any mechanisms. A compliant TOE must ensure the specified requirements are included in the FIPS 40-2 validation.

Requirements with cryptographic functionality not FIPS-approved:

The PP requires cryptographic functionality for key establishment for which there is currently no *FIPS-approved* key establishment techniques at this time.[2] The CMVP program allows these cryptographic functions to be implemented in a *FIPs-validated module running in FIPS-approved mode.* These requirements are specified in the PP using the terminology *FIPS-supported* or *non-FIPs* to

[2] While Annex D cites ANSI X9.17 for symmetric key establishment, this standard has since been rescinded and therefore not appropriate to meet the requirements for the PP.

specify whether they are implemented in a *FIPs-validated module running in FIPS-approved mode* or not, respectively. The ST author will select the option that correctly reflects the implementation. The distinction between *FIPS-supported* or *non-FIPS* is important to both clarify the implementation in the ST, and for considering the methodology for evaluation.

There is one requirement in this class that is an exception. This requirement is FCS_CKM_SYM_EXP.1, selection Cryptographic Key Establishment using Automated Loading, regarding key error detection and directly attached key devices. The requirement may be implemented outside of the definition of the cryptographic module. It is included in this class for clarity since it is part of key management. For this requirement the term *TSF* when the functionality is implemented outside of a cryptographic module.

Addressing the evolving list of *FIPS-approved* cryptographic functionality:

The list of *FIPS-approved* crypto functions changes as the CMVP program evolves. The requirements address this in the following manner:

- the FCS_BCM requirement is written to de-couple the required cryptographic functions from its status regarding FIPS validation. FCS_BCM applies for all *FIPS-approved* cryptographic functions that a compliant TOE must implement.
- The ST assignments/selection for requirements with cryptographic functionality not FIPS-approved includes the selection *FIPS-approved* which is to be used when the status of the cryptographic function has changed to be a *FIPS-approved* standard.

It's important to note to vendors and end users that any IT entity that is used to protect National Security Information, and employs cryptography as a protection mechanism, will require the TOE's key management techniques to be approved by NSA when the TOE is fielded.

FCS_BCM_EXP.1 Baseline Cryptographic Module

FCS_BCM_EXP.1.1 - All cryptographic functions implemented by the TOE that are *FIPS-approved* cryptographic functions shall be implemented in crypto module that is FIPS PUB 140-2 validated, and perform the specified cryptographic functions in a FIPS-approved mode of operation.

FCS_BCM_EXP.1.2 - All FIPS-validated cryptographic modules implemented in the TSF shall have a minimum overall Security Level 1 and meet Security Level 3 for the following: cryptographic module ports and interfaces; roles, services and authentication; cryptographic key management, and design assurance.

FCS_CKM.1 Cryptographic Key Generation (for symmetric keys using RNG)

FCS_CKM.1.1 **Refinement**: The **FIPS-validated cryptomodule** shall generate **symmetric** cryptographic keys [using a FIPS-Approved Random Number Generator**]** [for all key sizes] that meet the following: [one of the standards defined in Annex C to FIPS 140-2].

Application Note: This requirement specifies that the FIPS-validated cryptomodule

must be able to generate the AES keys, although nothing prevents externally-generated keys from being used as well (as long as the requirements in FCS_CKM_SYM_EXP.2 are met). Annex C to FIPS 140-2 defines FIPS-Approved random number generation algorithms. Each of the algorithms is defined in an associated standard listed in the Annex. The actual key size will be determined by the algorithm that uses the key; see FCS_COP_EXP.2.

FCS_CKM_SYM_EXP.1 Cryptographic Key Establishment for AES symmetric keys

Application Note: This PP requires that compliant TOEs be able to generate symmetric key (FCS_CKM.1); it also requires that symmetric key be able to be established either through a protocol exchange (e.g., Diffie-Hellman), or manual or automated input/output.

FCS_CKM_SYM_EXP.1.1 – The cryptomodule shall provide the following [selection: FIPS-approved, FIPS-supported security function, non-FIPS-supported security function, none] cryptographic key establishment technique(s) for symmetric keys:

Application Note: For the selection above, the ST writer should select "FIPS-supported security function" if the key establishment technique is implemented in a FIPS-validated cryptomodule running in a FIPS-approved mode of operation. If manual or automated loading is selected, the functionality must be implemented in a FIPS-validated module but the functionality is not cryptographic in nature (that is, no cryptographic algorithm is being exercised), but rather the functionality is present in the implementation of a FIPS-approved security function. In this case, "none" should be selected. In all other cases, select non-FIPS-supported security function. If multiple key establishment techniques are specified, FCS_CKM_SYM_EXP.1 should be iterated appropriately.

[selection:

a) Cryptographic Key Establishment using Discrete Logarithm Key Agreement that meets the following:

Application Note: This element of the top-level selection applies to automated key agreement schemes where an exchange occurs between the TOE and another IT entity that results in both entities having the same secret key without ever having passed that key between the two entities. This is in contrast to key transport schemes, where key is actually passed between two IT entities. This is also distinct from key loading, where the user is either directly inputting or receiving key, or an automated device (token, PC card, etc.) is inputting or receiving key.

a) The cryptomodule shall provide the capability to act as the initiator or responder (that is, act as Party U or Party V as defined in the standard) to agree on cryptographic keys of all sizes using the [selection: dhStatic, dhEphem, dhOneFlow, dhHybrid1, dhHybrid2, dhHybridOneFlow, MQV1, MQV2] key agreement scheme where domain parameter p is a prime of [assignment: size of

prime “p” in number of bits that is 3072 or greater] and domain parameter q is a prime of [assignment: size of prime “q” in number of bits that is 256 or greater], and that conforms with ANSI X9.42-2001, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography.

Application Note: It should be noted that the actual key size of the symmetric key agreed to when using this scheme will be a function of the algorithm that will be using the key, as specified in FCS_COP_EXP.2.

In the selection in paragraph a), one or more of the schemes should be chosen by the ST author, based on what schemes the TOE implements. Note that the requirement is for the cryptomodule to be able to act as either party (as detailed in the standard) for the chosen scheme(s).

The two assignments are used to specify the number of bits used for the domain parameters p and q (which are primes). The requirement above indicates that p must be a prime of at least 3072 bits, while q must be a prime of at least 256 bits. The ST author should fill in the appropriate number of bits based on the implementation. This applies if the implementation generates its own domain parameters, or if it obtains the domain parameters in some other way (e.g., hard-coded, obtained from an outside authority).

b) The cryptomodule shall conform to the standard using a FIPS-approved MAC function, a FIPS-approved Random Number generation function, and a FIPS-approved Hashing function.

c) The choices and options used in conforming to the key agreement scheme(s) are as follows: [assignment: options that the cryptomodule implements when implementing the selected key agreement schemes, including options for any prerequisite or dependant functions (e.g., domain parameter generation and validation).];

Application Note: In the X9.42-2001 standard there are several sections that are marked “optional”, or where a choice is given. Choices are, for example, how the domain parameters are obtained (generated or obtained from some other entity). Another example is the key derivation function that is implemented. ST authors should use the assignment to provide sufficient information so that 1) it is possible to test the implementation of the function in a repeatable fashion, and 2) readers (consumers) of the ST understand exactly what is done by the key agreement schemes implemented. The ST author should ensure that all of the prerequisite options/choices, as well as choices/options in dependant functions, are covered in the assignment.

- Cryptographic Key Establishment using Elliptic Curve Key Agreement that meets the following:

Application Note: This element of the top-level selection applies to automated key agreement schemes where an exchange occurs between the TOE and another IT entity that results in both entities having the same secret key without ever having passed that key between the two entities. This is in contrast to key transport schemes, where key is actually passed between two IT entities. This is also distinct from key loading, where the user is either directly inputting or receiving key, or an automated device (token, PC card, etc.) is inputting or receiving key.

a) The cryptomodule shall provide the capability to act as the initiator or responder (that is, act as Party U or Party V as defined in the standard) to agree on cryptographic keys of all sizes using the [selection: Ephemeral Unified Model, 1-Pass Diffie-Hellman, Static Unified Model, Combined Unified Model with Key Confirmation, 1-Pass Unified Model, Full Unified Model, Full Unified Model with Key Confirmation, Station-to-Station, 1-Pass MQV, Full MQV, Full MQV with Key Confirmation] key agreement scheme using Elliptic Curves with the order of the base point being a [assignment: size of the order of the base point "n" in number of bits that is 256 or greater]-bit value, and conforms to ANSI X9.63-2001, Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Elliptic Curve Cryptography.

Application Note: In the selection in paragraph a), one or more of the schemes should be chosen by the ST author, based on what schemes the TOE implements. Note that the requirement is for the cryptomodule to be able to act as either party (as detailed in the standard) for the chosen scheme(s) where the schemes are asymmetric.

The assignment is used to specify the number of bits used for the domain parameter n, which is the order of the base point of the curve chosen (the standard uses "n" to denote this value). The requirement above indicates that n must be at least a 256-bit value. The ST author should fill in the appropriate number of bits based on the implementation. This applies if the implementation generates its own domain parameters, or if it obtains the domain parameters in some other way (e.g., hard-coded, obtained from an outside authority).

b) The cryptomodule shall conform to the standard using a FIPS-approved MAC function, a FIPS-approved Random Number generation function, and a FIPS-approved Hashing function.

c) The choices and options used in conforming to the key transport scheme(s) are as follows: [assignment: options that the cryptomodule implements when implementing the selected key transport schemes, including options for any prerequisite or dependant functions (e.g., domain parameter generation and validation).];

Application Note: In the X9.63-2001 standard there are several sections that are marked "optional", or where a choice is given. Choices are, for example, in the domain parameter generation and validation section (Section 5.1) where domain parameters can be generated over F_p or over F_2^m. Another example is the Diffie-Hellman primitive (Standard or Modified) that is implemented. ST authors should use the assignment to provide sufficient information so that 1) it is possible to test the implementation of the function in a repeatable fashion, and 2) readers (consumers) of the ST understand exactly what is done by the key agreement schemes implemented. The ST author should ensure that all of the prerequisite options/choices, as well as choices/options in dependant functions, are covered in the assignment.

- Cryptographic Key Establishment using Key Transport that meets the following:

Application Note: This element of the top-level selection applies to automated key transport schemes where key is exchanged between the TOE and another IT entity. This is in contrast to key agreement schemes, where key is determined based on shared public information between two IT entities. This is also distinct from key loading, where the user is either directly inputting or receiving key, or an automated device (token, PC card, etc.) is inputting or receiving key.

b) The cryptomodule shall provide (act as the initiator) and accept (act as the responder) cryptographic keys to/from another IT Entity using the [selection: 1-Pass Transport Scheme; 3-Pass Transport Scheme; both the 1-Pass and 3-Pass Transport Schemes] using Elliptic Curves with the order of the base point being a [assignment: size of modulus "n" in number of bits that is 256 or greater]-bit value in a manner that conforms with ANSI X9.63-2001, Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Elliptic Curve Cryptography.

Application Note: In the selection in paragraph a), one or more of the schemes should be chosen by the ST author, based on what schemes the TOE implements. Note that the requirement is for the cryptomodule to be able to act as either party (as detailed in the standard) for the chosen scheme(s).

The assignment is used to specify the number of bits used for the domain parameter n, which is the order of the base point of the curve chosen (the standard uses "n" to denote this value). The requirement above indicates that n must be at least a 256-bit value. The ST author should fill in the appropriate number of bits based on the implementation. This applies if the implementation generates its own domain parameters, or if it obtains the domain parameters in some other way (e.g., hard-coded, obtained from an outside authority).

c) The cryptomodule shall conform to the standard using a FIPS-approved MAC function, a FIPS-approved Random Number generation function, and a FIPS-approved Hashing function.

d) The choices and options used in conforming to the key transport scheme(s) are as follows: [assignment: options that the cryptomodule implements when implementing the selected key transport schemes, including options for any prerequisite or dependant functions (e.g., domain parameter generation and validation).];

Application Note: In the X9.63-2001 standard there are several sections that are marked "optional", or where a choice is given. Choices are, for example, in the domain parameter generation and validation section (Section 5.1) where domain parameters can be generated over F_p or over F_2^m. Another example is the Diffie-Hellman primitive (Standard or Modified) that is implemented. ST authors should use the assignment to provide sufficient information so that 1) it is possible to test the implementation of the function in a repeatable fashion, and 2) readers (consumers) of the ST understand exactly what is done by the key agreement schemes implemented. The ST author should ensure that all of the prerequisite options/choices, as well as choices/options in dependant functions, are covered in the assignment.

- Cryptographic Key Establishment using Manual Methods

Application Note: This element of the top-level selection applies to the case where a human is either typing key into the cryptomodule, or the cryptomodule is outputting key to a display, for instance. The distinguishing feature is that the transaction is between a human and the cryptomodule, and ***not*** *between the cryptomodule and another IT device or IT media.*

a) The FIPS-validated cryptomodule shall be able to accept as input and be able to output in the following circumstances [assignment: circumstances under which the cryptomodule will output a key] cryptographic keys in accordance with FIPS-compliant Key Management techniques that meet the FIPS 140-2 Key Management Security Level 3, Key Entry and Output;

Application Note: The ST author should use the assignment to detail the conditions under which key is output from the cryptomodule (for example, only during a certain type of key generation activity).

Note that the phrase "FIPS-compliant Key Management techniques" refer to techniques that meet the FIPS 140-2 Key Management requirements for Key Entry and Output at security level 3.

Note that this requirement mandates that cryptomodules in the TSF have the ability to perform manual key input/output, and that this capability has been through the FIPS validation process.

- Cryptographic Key Establishment using Automated Methods

Application Note: This element of the top-level selection applies to automated key loading device. In the case where key is being transferred from the device to the cryptomodule the key is being "input". In the case where the key is being transferred from the cryptomodule to the device (for instance, a CA loading a user's private key into a token device) the key is being "output."

a) The FIPS-validated cryptomodule shall be able to accept as input and be able to output in the following circumstances [assignment: circumstances under which the cryptomodule will output a key] cryptographic keys using key management techniques that meet the following:

Application Note: The ST author should use the assignment to detail the conditions under which key is output from the cryptomodule (for example, only during a certain type of key generation activity).

- The TSF shall provide the capability to directly attach a key device by [selection: internal bus, serial port, USB port, audio device, [assignment: other non-network physical device]];

Application note: An example of a device attached by an internal bus would be a floppy device used for keys transported on floppy disks. Note that this requirement does not require that the device drivers be part of the cryptographic module.

- The [selection: FIPS-validated cryptomodule, TSF] shall perform key error detection scheme on keys input via electronic methods using [selection: parity check, [assignment: other key error detection scheme]]; and

Application Note: In the first selection, the ST should indicate whether the key error detection scheme is performed prior to the key reaching the cryptomodule (in which case the selection should be "TSF") or is performed by the cryptomodule. For instance, if the device is attached to a USB port and the USB driver (that is not part

of the cryptomodule) performs a parity check of the data coming off of the device, then the selection should be "none" since the USB driver is not part of the cryptomodule. However, if the USB driver performed no check and the cryptomodule, once it was passed the key by the driver, performed the check, then "FIPS-validated cryptomodule" should be chosen.

In the second selection, the ST author should indicate what error detection scheme is employed. The requirement above refers to errors in parity or structure of the key; it does not necessarily require checks on key "goodness", length, format, etc.

- FIPS 140-2 Key Management Security Level 3, Key Entry and Output.

Application Note: Note that this requirement mandates that cryptomodules in the TSF have the ability to perform automated key input/output, and that this capability has been through the FIPS validation process.

]

Application Note: The ST author selects one or more of the identified methods (i.e., the two key agreement schemes, key transport, manual loading or automated loading) used to establish cryptographic keys in the TOE.

FCS_CKM_ASYM_EXP.1 Cryptographic Key Entry for Digital Signature/verification private keys

Application Note: This PP requires that compliant TOEs be able to generate public/private key pairs in accordance with the chosen digital signature algorithm specified in FCS_COP_EXP. 3. In addition, it also requires that a private key be able to be entered via manual or automated methods.

FCS_CKM_ASYM_EXP.1.1 – The FIPS-validated cryptomodule shall provide the following cryptographic key entry technique(s) for the private key used for the asymmetric algorithm [assignment: cryptographic operation selected in FCS_COP_EXP.3]:

Application Note: Multiple key entry techniques available for a single FCS_COP_EXP.3 may be presented as a list in this requirement, however, if there are multiple key entry techniques to support multiple FCS_COP_EXP.3 cryptographic functions, then FCS_CKM_ASYM_EXP.1 should be iterated appropriately.

[selection*:*

- Cryptographic Key Establishment using Manual Methods

*Application Note: This element of the top-level selection applies to the case where a human is typing key into the cryptomodule. The distinguishing feature is that the transaction is between a human and the cryptomodule, and **not** between the cryptomodule and another IT device or IT media.*

a) The FIPS-validated cryptomodule shall be able to accept as input cryptographic keys in accordance with FIPS-compliant Key Management techniques that meet the FIPS 140-2 Key Management Security Level 3, Key Entry and Output;

Application Note: Note that the phrase "FIPS-compliant Key Management techniques" refer to techniques that meet the FIPS 140-2 Key Management requirements for Key Entry and Output at security level 3.

Note that this requirement mandates that cryptomodules in the TSF have the ability to perform manual key input for the private key, and that this capability has been through the FIPS validation process.

- Cryptographic Key Establishment using Automated Methods

Application Note: This element of the top-level selection applies to automated/electronic key loading device. In the case where key is being transferred from the device to the cryptomodule the key is being "input".

a) The FIPS-validated cryptomodule shall be able to accept as input cryptographic keys using key management techniques that meet the following:

- The TSF shall provide the capability to directly attach a key device by [selection: internal bus, serial port, USB port, audio device, [assignment: other non-network physical device]];

Application note: An example of a device attached by an internal bus would be a floppy device used for keys transported on floppy disks. Note that this requirement does not require that the device drivers be part of the cryptographic module.

- The [selection: FIPS-validated cryptomodule, TSF] shall perform key error detection scheme on keys input via electronic methods using [selection: parity check, [assignment: other key error detection scheme]]; and

Application Note: In the first selection, the ST should indicate whether the key error detection scheme is performed prior to the key reaching the cryptomodule (in which case the selection should be "TSF") or is performed by the cryptomodule. For instance, if the device is attached to a USB port and the USB driver (that is not part of the cryptomodule) performs a parity check of the data coming off of the device, then the selection should be "none" since the USB driver is not part of the cryptomodule. However, if the USB driver performed no check and the cryptomodule, once it was passed the key by the driver, performed the check, then "FIPS-validated cryptomodule" should be chosen.

In the second selection, the ST author should indicate what error detection scheme is employed. The requirement above refers to errors in parity or structure of the key; it does not necessarily require checks on key "goodness", length, format, etc.

- FIPS 140-2 Key Management Security Level 3, Key Entry and Output.

Application Note: Note that this requirement mandates that cryptomodules in the TSF have the ability to perform automated key input, and that this capability has been through the FIPS validation process.

]

Application Note: The ST author selects one or more of the identified methods (i.e., manual loading or automated loading) used to establish asymmetric cryptographic keys in the TOE.

FCS_CKM.4 Cryptographic Key Destruction

FCS_CKM.4.1 - **Refinement**: The TSF shall destroy cryptographic keys in accordance with a [cryptographic key **zeroization** method**]** that meets the following:

a) [The Key Zeroization Requirements in FIPS PUB 140-2 Key Management Security Level 3;

b) Zeroization of all private cryptographic keys, plaintext cryptographic keys and all other critical cryptographic security parameters shall be immediate and complete; and

c) The zeroization shall be executed by overwriting the key/critical cryptographic security parameter storage area three or more times with an alternating pattern.

d) The TSF shall overwrite each intermediate storage area for private cryptographic keys, plaintext cryptographic keys, and all other critical security parameters three or more times with an alternating pattern upon the transfer of the key/CSPs to another location.]

Application note: Item d applies to locations that are used when the keys/parameters are copied during processing, and not to the locations that are used for storage of the keys, which are specified in items b and c. The temporary locations could include memory registers, physical memory locations, and even page files and memory dumps.

FCS_COP_EXP.2 Cryptographic Operation (Encryption/Decryption using AES)

FCS_COP_EXP.2.1 A cryptomodule shall perform encryption and decryption using the FIPS-Approved Security Function AES algorithm operating in [selection: one or more of ECB, CBC, OFB, CFB1, CFB8, CFB128, CTR] mode(s) supporting key sizes of [selection: one or more of 128 bits, 192 bits, 256 bits].

Application note: Item d applies to locations that are used when the keys/parameters are copied during processing, and not to the locations that are used for storage of the keys, which are specified in items b and c. The temporary locations could include memory registers, physical memory locations, and even page files and memory dumps.

Note that this requirement applies to all cryptomodules in the TSF, whether they are FIPS-validated or not. As a practical matter, FIPS-validated cryptomodules will have to have the above functionality implemented and tested as part of the CMVP validation so that the fact that the key destruction is being performed as specified above in a FIPS-approved mode of operation can be established.

FCS_COP_EXP.3 Cryptographic Operation (Digital Signature Generation/Verification)

FCS_COP_EXP.3.1 A cryptomodule shall perform digital signature generation and verification using the FIPS-Approved Security Function [selection:

- rDSA

Application Note: This top-level selection indicates that the digital signatures will be calculated using the rDSA algorithm specified in X9.31-1998, as implemented in a FIPS-validated cryptomodule.

a) The cryptomodule shall implement rDSA with a modulus size of [assignment: size of modulus "n" in number of bits that is 2048 bits or greater] in a manner that

conforms to ANSI X9.31-1998, Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA).

Application Note: The ST author should fill in the assignment with the number of bits the module uses for its modulii. Note that in order to meet the requirement modulii must be at least 2048 bits.

b) The choices and options used in conforming to the X9.31-1998 are as follows: [assignment: options that the cryptomodule implements when implementing the signature generation and validation functions, including options for any prerequisite or dependant functions (e.g., key generation).];

Application Note: In the X9.31-1998 standard there are several sections that are marked "optional", or where a choice is given. For instance, the public verification exponent "e" can be fixed or randomly generated. Another instance is that the procedure in section 4.1.2.1 can be followed to generate the primes p and q, or another procedure followed as long as the primes generated meet the conditions in section 4.1.2. The goal of the assignment is to provide sufficient information such that 1) it is possible to test the implementation of the function in a repeatable fashion, and 2) readers (consumers) of the ST understand exactly what is done by the rDSA implementation. The ST author should ensure that all of the prerequisite options/choices, as well as choices/options in dependant functions, are covered in the assignment.

- ECDSA

Application Note: This top-level selection indicates that the digital signatures will be calculated using the ECDSA algorithm specified in X9.62-1998, as implemented in a FIPS-validated cryptomodule.

a) The FIPS-validated cryptomodule shall implement ECDSA where the order of the base point is a [assignment: size of the order of the base point "n" in number of bits that is 256 or greater]-bit value, and where the algorithm conforms with ANSI X9.62-1998, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA).

Application Note: The assignment is used to specify the number of bits used for the domain parameter n, which is the order of the base point of the curve chosen (the standard uses "n" to denote this value). The requirement above indicates that n must be at least a 256-bit value. The ST writer should fill in the appropriate number of bits based on the implementation. This applies if the implementation generates its own domain parameters, or if it obtains the domain parameters in some other way (e.g., hard-coded, obtained from an outside authority).

b) The choices and options used in conforming to X9.62-1998 are as follows: [assignment: options that the TSF implements when implementing the signature

generation and validation functions, including options for any prerequisite or dependant functions (e.g., domain parameter generation and validation)].]

Application Note: In the X9.62-1998 standard there are several sections that are marked "optional", or where a choice is given. Choices are, for example, in the domain parameter generation and validation section (Section 5.1) where domain parameters can be generated over F_p or over F_2^m. Public Key validation is an example of an optional part of the standard. ST writers should use the assignment to provide sufficient information such that 1) it is possible to test the implementation of the function in a repeatable fashion, and 2) readers (consumers) of the ST understand exactly what is done by the key transport schemes implemented. The ST author should ensure that all of the prerequisite options/choices, as well as choices/options in dependant functions, are covered in the assignment.

FCS_COP_EXP.5 Cryptographic Operation (Random Number Generation)

FCS_COP_EXP.5.1 The TSF shall perform all Random Number Generation used by the cryptographic functionality of the TSF, as well as all SFRs that require random numbers, using a FIPS-approved Random Number Generator implemented in a FIPS-approved cryptomodule running in a FIPS-approved mode.

Application Note: Whenever a referenced standard calls for a random number generation capability, this requirement specifies the subset of random number generators (those that are FIPS-validated) that are acceptable. Note that the RNG does not have to be implemented in the cryptomodule that is performing the cryptographic operation. This also requires that if implementation of an SFR requires a number to be randomly generated, then a RNG in a FIPS-validated cryptomodule is used. For example, if an SFR specified that TCP sequence numbers were to be randomly generated in order to counter TCP session hijacking attempts, the TCP sequence numbers would have to be randomly generated using the functionality in a FIPS-validated cryptomodule. On the other hand, if the TSF randomly generated temporary filenames (and this capability was unrelated to any SFR in the ST) then any RNG could be used. Note that this requirement is not calling for the RNG functionality to be made generally available (e.g., to untrusted users via an API).

FCS_COP_EXP.6 Cryptographic Operation (Cryptographic Hashing Function)

FCS_COP_EXP.6.1 The TSF shall perform all Cryptographic Hashing Functions used by other cryptographic functionality of the TSF using a FIPS-approved Cryptographic Hashing Function implemented in a FIPS-approved cryptomodule running in a FIPS-approved mode.

Application Note: Whenever a referenced standard calls for a cryptographic hashing capability (e.g., SHA-1), this requirement specifies the subset of cryptographic

hashing functions (those that are FIPS-validated) that are acceptable. Note that the hashing function does not have to be implemented in the cryptomodule that is performing the cryptographic operation. Also note that this requirement is not calling for the hashing functionality to be made generally available (e.g., to untrusted users via an API).

5.1.3 User data protection (FDP)

FDP_IFC.1(1) Subset information flow control (unauthenticated policy)

FDP_IFC.1.1(1) - The TSF shall enforce the [UNAUTHENTICATED INFORMATION FLOW SFP] on

- [source subject: TOE interface on which information is received;
- destination subject: TOE interface to which information is destined;
- information: network packets; and
- operations: pass information,

Application Note: In a firewall, the central issue is that there are two "subjects" (the sender of the packet (information) and the receiver of the packet) neither of which are under the control of the TOE. In order to use the FDP_IF requirements, we associate the potential set of subjects with a firewall interface. This makes sense because an administrator is able to determine what sets of IP addresses (for example) are associated with each of the physical firewall interfaces (assuming no other "backdoor" connectivity). Associating this potential set of subjects with an interface also allows the specification of subject attributes to be associated with something that is actually part of the TOE (the physical interface), as well as allow FDP_IFF.1.2-NIAP-0417 to be written so that it actually makes sense.*

Note that "operations" also is different from an operating-system-centric world because there is only one operation that the subjects really want: that the information is passed through the firewall.

FDP_IFF.1-NIAP-0417(1) Simple security attributes (unauthenticated policy)

FDP_IFF.1.1-NIAP-0417(1) - The TSF shall enforce the [UNAUTHENTICATED INFORMATION FLOW SFP] based on the following types of subject and information security attributes:

a) [Source subject security attributes:

- set of source subject identifiers; and
- [selection: [assignment: other subject security attributes], none].

b) Destination subject security attributes:

- Set of destination subject identifiers; and
- [selection: [assignment: other subject security attributes], none].

Application Note: For the subjects, the administrator knows the set of identifiers that can be associated with the physical firewall interfaces; therefore, they are not "presumed" identifiers. The term "identifiers" was used instead of "addresses" to allow for technologies that are not address-based (e.g., circuit identifiers instead of source and destination addresses).

The ST author should specify other attributes that are used to identify the source and destination subject sets, based on the technology implemented by the TOE.

c) Information security attributes:

- presumed identity of source subject[3];
- identity of destination subject;
- transport layer protocol;
- source subject service identifier;
- destination subject service identifier (e.g., TCP or UDP destination port number);

Application Note: The transport layer protocol is what is specified in the 8-bit protocol field in the IP header (e.g., this would include ICMP and is not limited to TCP or UDP). The concept of a "service identifier" may differ depending on the networking stack used; the intent is to specify a service that is above the network and transport layers in the protocol stack. A "service" in the IP stack would be NTP, Trivial File Transfer Protocol (TFTP), etc.

- [selection: [assignment: other information security attributes], none].

Application Note: Not all of the above security attributes will exist in all network packets. However, the TOE's ruleset allows the Security Administrator to select and filter on any of the above security attributes as part of the policy decision. The intent is that if a network packet includes any of the above security attributes, those attributes will be used in the policy decision. The ST author should fill in the assignment all attributes that the Security Administrator is able to specify when

[3] The TOE can make no claim as to the real identity of any source subject; the TOE can only suppose that such identities are accurate. Therefore, a "presumed identity" is used to identify source subjects. Note, however, that the TOE can ensure that the identity is included in the set that is associated with the interface (see FDP_IFF.1.6(1)).

creating the firewall rules.

- Stateful packet attributes: [selection: for IP-based network stacks:
  - Connection-oriented protocols:
    - sequence number;
    - acknowledgement number;
    - Flags:
      - SYN;
      - ACK;
      - RST;
      - FIN; and
    - [selection: [assignment: other information security attributes], none].
  - Connectionless protocols:
    - source and destination network identifiers;
    - source and destination service identifiers;
    - [selection:[assignment: other information security attributes], none];,

for non-IP-based network stacks: [assignment: information security attributes]].

Application Note: The stateful packet attributes are not specified in the ruleset as are the other security attributes. These attributes are intended to be used in FDP_IFF.1.3-NIAP-0417(1) as part of the stateful packet inspection. A TOE conformant to this PP keeps state about a connection (e.g., a TCP connection) or pseudo-connection (e.g., UDP stream) and uses that information in determining whether to permit information to flow. If a compliant TOE uses an IP based network stack (including IP running on top of another protocol, such as Asynchronous Transfer Method (ATM)), then the first selection is made. If the TOE uses a network stack that is not IP-based (e.g., ATM without IP) then the ST author uses the second selection and fills in the assignment with the attributes that provide a commensurate level of confidence for the protocols employed that network packets can be correctly associated with a connection.

FDP_IFF.1.2-NIAP-0417(1) - Refinement: The TSF shall permit an information flow between a **source subject and a destination subject** via a controlled operation if the following rules hold:

- **[**the presumed identity of the source subject is in the set of source subject identifiers;

- the identity of the destination subject is in the set of source destination identifiers;

- the information security attributes match the attributes in an information flow policy rule (contained in the information flow policy ruleset defined by the Security Administrator) according to the following algorithm [assignment: algorithm used by the TOE to match information security attributes to information flow policy rules]; and

- the selected information flow policy rule specifies that the information flow is to be permitted].

Application Note: In a firewall, the administrator specifies information flow policy rules that contain information security attribute values (or wildcards that "stand" for multiple values of the same type; e.g., 127..*.* would represent any IP address that begins with "127"), and associate with that rule an action that permits the information flow or disallows the information flow. When a packet arrives at the source interface, the information security attribute values of the packet are compared to each information flow policy rule by some TOE-specified algorithm, and when a match is found the action specified by that rule is taken. Since wildcards would allow the specific attributes in a packet to potentially match more than one rule, the ST author needs to fill in the assignment with the algorithm the TOE uses to find a matching a rule. This could be "first match", "most specific match", or some more elaborate description.*

FDP_IFF.1.3-NIAP-0417(1) - The TSF shall enforce the [following:

- fragmentation rule:

  - prior to applying the information policy ruleset, the TOE completely reassembles fragmented packets;

- stateful packet inspection rules:

  - whenever a packet is received that is not associated with an allowed established session (e.g., the SYN flag is set without the ACK flag being set), the information flow policy ruleset, as defined in FDP_IFF.1.2-NIAP-0417(1), is applied to the packet;

  - otherwise, the TSF associates a packet with an allowed established session using the stateful packet attributes].

Application Note: This requirement has two distinctive rules that are applied. The

first rule ensures that the TOE reassembles packets before applying the policy rules. The TOE ensures that fragments are handled properly and the TOE will drop any malformed packets (e.g., duplicate fragments, invalid offsets) and eliminates the security concern of fragments being received out of order at the target host.

The second rule, requires that the TOE maintains state for connection-oriented sessions and connectionless "pseudo" sessions. The TOE uses the stateful packet attributes to determine if a packet already belongs to a "session" that has been allowed by the TOE's ruleset. If a packet cannot be associated with a session, then the ruleset is applied. Connectionless sessions are subject to these rules and allow an IT entity to respond to a connectionless packet without having to specify a rule in the ruleset to explicitly allow the flow.

When a packet is received, usually "sanity" checks are made first (e.g., format and frame checks to make sure that the packet is well formed). If an address is all zeros (e.g., MAC address, Source IP address), the packet is discarded. If the packet passes the sanity checks, the TOE searches to see if the packet is associated with an existing session. If it is connectionless, the TOE may create a "pseudo session" to associate connectionless packets with a connection and therefore represent the connectionless data stream.

In an IP-based network stack, if a session already exists, the TCP packet's sequence number, acknowledgment number and flags (e.g., SYN, FIN) are checked to make sure that the packet really belongs to the session (e.g., an invalid sequence number can indicate a hijacked session). The ST author may include other security attributes (e.g., window size) if they so desire. If the checks pass, then the packet is allowed to pass. If the packet cannot be associated with an established session, the TOE's ruleset is applied to the packet.

Connection-less protocols (e.g., UDP) are included in the stateful inspection rules to allow for a "pseudo connection", which allows return traffic through the TOE without having to specify a rule in the TOE's ruleset.

FDP_IFF.1.4-NIAP-0417(1) - The TSF shall provide the following [the Security Administrator shall have the capability to view all information flows allowed by the information flow policy ruleset before the ruleset is applied].

Application Note: Some firewalls create additional rules as a side-effect of creating a rule; for example, a firewall may create a rule allowing an FTP data channel when a rule allowing FTP (control connections) is created. This requirement allows an administrator to view the entire ruleset so that they can identify such rules and confirm that the ruleset reflects the desired policy.

"before the rule set is applied" means that the administrator is able to view the entire rule set before it is put into use on the TOE. This gives the administrator the opportunity to address any errors or unintended flows.

FDP_IFF.1.5-NIAP-0417(1) - The TSF shall explicitly authorize an information flow based on the following rules: [none].

FDP_IFF.1.6-NIAP-0417(1) - The TSF shall explicitly deny an information flow based on the following rules:

a) [The TOE shall reject requests for access or services where the presumed source identity of the information received by the TOE is not included in the set of source identifiers for the source subject;

Application Note: The intent of this requirement is to ensure that a user cannot send packets originating on one TOE interface claiming to originate on another TOE interface.

b) The TOE shall reject requests for access or services where the presumed source identity of the information received by the TOE specifies a broadcast identity;

Application Note: A broadcast identity is one that specifies more than one host address on a network. It is understood that the TOE can only know the sub-netting configuration of networks directly connected to the TOE's interfaces and therefore can only be aware of broadcast addresses on those networks.

c) The TOE shall reject requests for access or services where the presumed source identity of the information received by the TOE specifies a loopback identifier;

d) The TOE shall reject requests in which the information received by the TOE contains the route (set of host network identifiers) by which information shall flow from the source subject to the destination subject].

FDP_IFC.1(2) Subset information flow control (unauthenticated TOE services policy)

FDP_IFC.1.1(2) - The TSF shall enforce the [UNAUTHENTICATED TOE SERVICES SFP] on

a) [source subject: TOE interface on which information is received;

b) destination subject: the TOE;

c) information: network packets; and

d) operations: accept or reject network packet].

Application Note: This policy is used to express how the TOE enforces rules concerning network traffic that is destined for the TOE, and the protocols that are allowed as specified in FIA_UAU.1(1). The intent of this iteration of the requirement is control how the TOE responds to network traffic destined for the TOE, this policy does not have to be enforced in the firewall ruleset (e.g., could be Security

Administrator configurable and TOE controlled via another mechanism).

Note that "operations" refers to the TOE accepting or rejecting the network packet, since the TOE is not technically always providing the "service". In the case of ARP, another machine (e.g., router on the same subnet) is providing an ARP "service" by providing updates to the TOE's routing tables.

FDP_IFF.1-NIAP-0417(2) Simple security attributes (unauthenticated TOE services policy)

FDP_IFF.1.1-NIAP-0417(2) - The TSF shall enforce the [UNAUTHENTICATED TOE SERVICES SFP] based on the following types of subject and information security attributes:

a) [Source subject security attributes:

- set of source subject identifiers; and
- [selection: [assignment: other subject security attributes], none].

b) Destination subject security attributes:

- TOE's network identifier; and
- [selection: [assignment: other subject security attributes], none].

Application Note: For the subjects, the administrator knows the set of identifiers that can be associated with the physical firewall interfaces; therefore, they are not "presumed" identifiers. The term "identifiers" was used instead of "addresses" to allow for technologies that are not address-based (e.g., circuit identifiers instead of source and destination addresses).

The ST author should specify other attributes that are used to identify the source and destination subject sets, based on the technology implemented by the TOE.

c) Information security attributes:

- presumed identity of source subject;
- identity of destination subject;
- transport layer protocol;
- source subject service identifier;
- destination subject service identifier (e.g., TCP or UDP destination port number); and

Application Note: Not all of the above security attributes will exist in all network packets. The intent is that if a network packet includes any of the above security attributes, those attributes will be used in the policy decision. The data link frame type identifies the type of data the data link header encapsulates (e.g., in the case of ARP, the frame type value is 0x0806). The transport layer protocol is what is specified in the 8-bit protocol field in the IP header (e.g., this would include ICMP (value of 1) and is not limited to TCP (value of 6) or UDP (value of 17)). The concept of a "service identifier" may differ depending on the networking stack used; the intent is to specify a service that may exist above the network and transport layers in the protocol stack. A "service" in the IP stack would be NTP, TFTP, etc.

- [selection: for an IP-based network stack: ICMP message type and code as specified in RFC 792, [selection: [assignment: other information security attributes associated with services identified in FAU_UAU.1(1)], none]; or for a non-IP-based network stack: [assignment: information security attributes]].

Application Note: For an IP-based network stack, the ICMP is to be controlled at the message type and code level. The ST author should fill in the first assignment with the attributes associated with services provided by the TOE that the Security Administrator is able to specify when configuring this policy. If no additional services are specified in FAU_UAU.1(1), the ST author should fill the selection with none. If the TOE uses a non-IP-based network stack, than the ST author makes the second selection and assigns attributes to the services identified in FAU_UAU.1(1).

FDP_IFF.1.2-NIAP-0417(2) – **Refinement**: The TSF shall permit an information flow between a **source** subject and **the TOE** via a controlled operation if the following rules hold:

- [the presumed identity of the source subject is in the set of source subject identifiers;
- the identity of the destination subject is the TOE;
- the information security attributes match the attributes in an information flow control policy according to the following algorithm [assignment: algorithm used by the TOE to match information security attributes to information flow control policy].

Application Note: This bullet is dependent on the ST's implementation and may have an assignment of none, if the implementation of this policy does not use the TOE ruleset (e.g., another mechanism is used to control the information flow to/from the TOE).

FDP_IFF.1.3-NIAP-0417(2) - The TSF shall enforce the [following rules:

- The TOE shall allow source subjects to access TOE services [selection: for an IP-based network stack: ICMP, [selection: [assignment: list of other network

services provided by the TOE, consistent with FIA_UAU.1(1)], none]; or for non-IP-based network stacks: [assignment: list of network services provided by the TOE, consistent with FIA_UAU.1(1)]] without authenticating those source subjects; and

- The TOE shall allow the list of services specified immediately above to be enabled (become available to unauthenticated users) or disabled (become unavailable to unauthenticated users)].

Application Note: The intent of this requirement (first bullet) is to allow users to access services such as ICMP Echo (ping) without authentication. However, since some sites may not want to allow this capability, the second bullet was added so that an administrator (see FMT_MOF.1(6)) can restrict the services available.

The ST author should fill in the assignment in the first bullet with a list of services that the firewall provides that can be accessed without authentication by the user, and make sure that this list is the same as is provided in FIA_UAU.1.1(1)].

FDP_IFF.1.4-NIAP-0417(2) - The TSF shall provide the following [the Security Administrator shall have the capability to view all information flows allowed by this information flow control policy before the policy is applied].

Application Note: The intent here is to provide the Security Administrator the capability to see what information flow controls will be applied to the TOE before those controls are activated. This gives the Security Administrator the opportunity to address any errors or unintended TOE interactions with users. In the case of this policy, information flow is between a network device and the TOE.

FDP_IFF.1.5-NIAP-0417(2) - The TSF shall explicitly authorize an information flow based on the following rules: [none].

FDP_IFF.1.6-NIAP-0417(2) - The TSF shall explicitly deny an information flow based on the following rules:

- [The TOE shall reject requests for access or services where the presumed source identity of the information received by the TOE is not included in the set of source identifiers for the source subject;

- The TOE shall reject requests for access or services where the presumed source identity of the information received by the TOE specifies a broadcast identity;

Application Note: A broadcast identity is one that specifies more than one host on a network. It is understood that the TOE can only know the sub-netting configuration of networks directly connected to the TOE's interfaces and therefore can only be aware of broadcast addresses on those networks.

- The TOE shall reject requests for access or services where the presumed source identity of the information received by the TOE specifies a loopback identifier; and

- The TOE shall reject requests in which the information received by the TOE contains the route (set of host network identifiers) by which information shall flow from the source subject to the TOE].

FDP_RIP.2 Full residual information protection

FDP_RIP.2.1 - 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.

Application Note: One aspect of this requirement is to ensure packets do not contain residual information that may be used in the padding of a packet.

5.1.4 Identification and authentication (FIA)

TOE security functions implemented by a probabilistic or permutational mechanism (e.g., password or hash function) are required (at EAL2 and higher) to include a strength of function claim. Strength of Function shall be demonstrated for the authentication mechanism used by the administrators to be SOF-medium, as defined in Part 1 of the CC. Specifically, the local authentication mechanism must demonstrate adequate protection against attackers possessing a moderate attack potential.

FIA_AFL.1-NIAP-0425 Authentication failure handling

FIA_AFL.1.1-NIAP-0425 - **Refinement**: The TSF shall detect when [a Security Administrator-configurable integer] **of** unsuccessful authentication attempts occur related to [administrators attempting to authenticate remotely and authorized IT entities].

Application Note: This requirement also does not apply to the local administrators, since it does not make sense to lock a local administrator's account in this fashion. This could be addressed by requiring a separate account for local administrators, which would be stated in the administrative guidance, or the TOE's authentication mechanism implementation could distinguish login attempts that are made locally and remotely.

FIA_AFL.1.2-NIAP-0425 – **Refinement:** When the defined number of unsuccessful authentication attempts has been met, the TSF shall [at the option of the Security Administrator prevent the remote administrators or authorized IT entity from performing activities that require authentication until an action is taken by the Security Administrator, or until a Security Administrator defined time period has elapsed].

FIA_ATD.1 User attribute definition

FIA_ATD.1.1 – **Refinement:** The TSF shall maintain the following list of security attributes belonging to **an authorized user**:

a) [user identifier(s):

- role;
- [selection: [assignment: Any security attributes related to a user identifier (e.g., certificate associated with the userid)], none]; and

b) [selection: [assignment: other user security attributes], none]].

Application Note: This requirement applies to authorized administrators and authorized IT entities. The intent is to allow multiple userids to be associated with a user. This allows a single human user to assume multiple roles, albeit requiring authentication as the userid associated with a given role. The intent is for a userid to only be associated with a single role, thus limiting the amount of damage if an administrative role is compromised.

FIA_UAU.1(1) Timing of authentication (for TOE services)

FIA_UAU.1.1(1) - The TSF shall allow [selection: for an IP-based network stack: ICMP, [selection: [assignment: list of other TOE-provided services], none]; or for a non-IP-based network stack: [assignment: list of TOE-provided services]] on behalf of the user to be performed before the user is authenticated.

FIA_UAU.1.2(1) - The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

Application Note: The ST author should fill in the assignment on the list of services provided by the TOE that are accessible to users without authentication. For an IP-based network stack ICMP is required. For a non-IP-based network stack the ST author chooses the second selection and fills in the assignment with TOE services, including a control-level protocol similar to ICMP for IP.

FIA_UAU.1 Timing of authentication

FIA_UAU.1.1 - The TSF shall allow [selection: for an IP-based network stack: ICMP, selection: [assignment: list of other TOE-provided services], non]; or for a non-IP-based network stack: [assignment list of TOE-provided services]] on behalf of the user to be performed before the user is authenticated.

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

Application Note: The assignment should be filled in with the types of traffic that can flow through the TOE without requiring authentication.

Explicit: FIA_UAU_EXP.2 Specified User authentication before any action

FIA_UAU_EXP.2.1- The TSF shall require the administrators and authorized IT entities to be successfully authenticated before allowing any other TSF-mediated actions on behalf of these authorized users.

Application Note: Although FIA_UAU.1.2(*) requires all other actions by users to be mediated, this requirement is levied to make the set of users required to authenticate to the TOE clear. Note that the authentication is required only when the specified user is performing a function related to the authentication; for instance, if a user that happens to be an administrator wants to utilize an unauthenticated service from the list in FIA_UAU.1.1(1), they are not required to authenticate to that service.

Explicit: Authentication Mechanism(FIA_UAU_EXP.5)

FIA_UAU_EXP.5.1 - The TSF shall provide a local authentication mechanism, [selection: [assignment: other authentication mechanism(s)], none] to perform user authentication.

Application Note: This explicit requirement is needed because there is no CC requirement (other than FIA_UAU.5) that requires the TSF provide authentication (it is implied by other FIA_UAU requirements, but not explicitly required).

The ST author could chose to fill in the assignment with any additional authentication mechanism such as a single-use authentication mechanism, or a mechanism that authenticates users by using a certificate. If an asymmetric algorithm is chosen, the TOE may rely upon a certificate authority server to obtain a user's certificate, and this server would be considered an authorized IT entity and IT environment requirements should be levied on this IT entity.

FIA_UID.2 User identification before any action

FIA_UID.2.1 - The TSF shall require each user to identify itself before allowing any other TSF-mediated actions on behalf of that user.

Application Note: All users, whether authenticated or not, will always be identified at least by a source network identifier. In the case of authenticated users (administrators and authorized IT entities) there will probably be a "userid".

FIA_USB.1 User-Subject Binding

FIA_USB.1.1 - **Refinement:** The TSF shall associate **all user** security attributes with subjects acting on behalf of that **authorized** user.

Application Note: User security attributes are defined in FIA_ATD.1.

5.1.5 Security management (FMT)

FMT_MOF.1(1) - Management of security functions behavior (TSF non-Cryptographic Self-test)

FMT_MOF.1.1(1) - The TSF shall restrict the ability to *modify the behavior of* the functions

- [TSF Self-Test (FPT_TST_EXP.4)]

to [the Security Administrator].

Application Note: "Modify the behavior" refers to specifying the interval at which the test periodically run, or perhaps selecting a subset of the tests to run.

FMT_MOF.1(2) - Management of security functions behavior (Cryptographic Self-test)

FMT_MOF.1.1(2) - The TSF shall restrict the ability to *enable, disable* the functions

- [TSF Self-Test (FPT_TST_EXP.5)]

to [the Cryptographic Administrator].

Application Note: The enabling or disabling of the cryptographic self-tests immediately after key generation.

FMT_MOF.1(3) Management of security functions behavior (audit and alarms)

FMT_MOF.1.1(3) - The TSF shall restrict the ability to *enable, disable, determine and modify the behavior of* the functions

- [Security Audit (FAU_SAR)] to [an Administrator].

FMT_MOF.1(4) Management of security functions behavior (audit and alarms)

FMT_MOF.1.1(4) - The TSF shall restrict the ability to *enable, disable, determine and modify the behavior of* the functions

- [Security Audit Analysis (FAU_SAA); and
- Security Audit (FAU_SEL)]

to [the Security Administrator].

Application Note: For the Audit function, enable and disable refer to the ability to enable or disable the audit mechanism as a whole. "Determine the behavior" means the ability to determine specifically what on the system is being audited, while "modify the behavior" means the ability to set or unset specific aspects of the audit

mechanism, such as what user behavior is audited, etc.

FMT_MOF.1(5) Management of security functions behavior (audit and alarms)

FMT_MOF.1.1(5) - The TSF shall restrict the ability to *enable, or disable* the functions

- [Security Alarms (FAU_ARP)]

to [the Security Administrator].

Application Note: This requirement ensures only the Security Administrator can enable or disable (turn on or turn off) the alarm notification function – messages and/or the audible alarm. As currently written, FAU_ARP.1 does not lend itself to behavior modification. If the ST author were to include additional functionality in FAU_ARP.1 (e.g., notify the administrator via a pager) then the ST author should consider adding, "modify the behavior" to this requirement.

FMT_MOF.1(6) Management of security functions behavior (available TOE-services for unauthenticated users)

FMT_MOF.1.1(6) - The TSF shall restrict the ability to *enable, disable* the functions

- [[selection: for an IP-based network stack: ICMP, [selection: [assignment: other defined services for which authentication is not required in (FIA_UAU.1(1))], none], or for a non-IP-based network stack:[assignment: defined services for which authentication is not required in (FIA_UAU.1(1))]]]

to [the Security Administrator].

Application Note: "Enable" refers to allowing a specific service to be specified as being one that is available to users on the network without those users first authenticating to the TOE. "Disable" refers to not allowing such a service to be available. This requirement, coupled with FIA_UAU.1, allows the Security Administrator to specify which TOE services are available to network users without authentication; if they choose, they can completely disable all such services so that unauthenticated users may only attempt to send traffic through the firewall. For the protocol required in FIA_UAU.1, this requirement defines the minimum level of control that must be provided to the Security Administrator. If the ST author provides additional services in FIA_UAU.1, then they should consider specifying the level of control the Security Administrator has with respect to those protocols in this requirement.

FMT_MOF.1(7) Management of security functions behavior (quota mechanism)

FMT_MOF.1.1(7) - The TSF shall restrict the ability to *determine the behavior of* the functions

- [Controlled connection-oriented resource allocation (FRU_RSA.1(2));
- an administrator-specified network identifier;
- set of administrator-specified network identifiers;
- administrator-specified period of time]

to [the Security Administrator].

Application Note: "determine the behavior of" refers to specifying the network identifier(s) that will be tracked using the FRU_RSA.1(2) requirement and the time period over which the quota limitations are enforced. Note that the specification of the actual quotas, while part of the resource allocation functionality, is done by FMT_MTD.2(2).

FMT_MSA.1 Management of security attributes

FMT_MSA.1.1 - The TSF shall enforce the [UNAUTHENTICATED INFORMATION FLOW SFP, UNAUTHENTICATED TOE SERVICES SFP] to restrict the ability to *manipulate* the security attributes [referenced in the indicated polices] to [the Security Administrator].

Application Note: The term "manipulate" is used to indicate that the security attributes in FDP_IFF.1.1-NIAP-0417 may be used to create additional "attributes" that can be used in specifying information flow policy rules (for example, a set of network identifiers that can be used as a "group"); this requirement restricts such capabilities to the Security Administrator.

It is important to note that the attributes associated with stateful packet inspection are not expected to be managed by the Security Administrator.

FMT_MSA.3-NIAP-0409(1) Static attribute initialization (ruleset)

FMT_MSA.3.1-NIAP-0409(1) – **Refinement:** The TSF shall enforce the [UNAUTHENTICATED INFORMATION FLOW SFP] to provide *restrictive* default values for the **information flow policy ruleset** that is used to enforce the SFP.

Application Note: "restrictive" in this case means that by default information is not allowed to flow (according to the referenced policies) unless an explicit rule in the information flow policy ruleset allows an information flow. By default, information is not allowed to flow.

FMT_MSA.3.2-NIAP-0409(1) - The TSF shall allow the [Security Administrator] to specify alternative initial values to override the default values when an object or information is created.

Application Note: Since a firewall ruleset typically does not provide multiple initial

default values for the rules (that is, there is generally only one "default" rule) this requirement may not apply for all TOEs. In TOEs that allow default values to be specified for individual rules, this requirement indicates that the specification must be done by the Security Administrator.

FMT_MSA.3-NIAP-0409(2) Static attribute initialization (services)

FMT_MSA.3.1-NIAP-0409(2) – **Refinement:** The TSF shall enforce the [UNAUTHENTICATED TOE SERVICES SFP] to provide *restrictive* default values for **the set of TOE services available to unauthenticated users.**

Application Note: Since FDP_IFF.1.3-NIAP-0417(2) allows the TOE to provide services to unauthenticated users, "restrictive" in this case indicates that such services are not available by default, and must be explicitly enabled by the Security Administrator.

FMT_MSA.3.2-NIAP-0409(2) - The TSF shall allow the [Security Administrator] to specify alternative initial values to override the default values when an object or information is created.

Application Note: This component was used to ensure that no services are provided to unauthenticated users by default, and that the Security Administrator has control over this list of services. FMT_MSA.3.2-NIAP-0409(2) might be used to allow the Security Administrator to allow a service to be enabled when the TOE is restarted, rather than having the service unavailable by default when the TOE boots up.

FMT_MTD.1(1) Management of TSF data (non-cryptographic, non-time TSF data)

FMT_MTD.1.1(1) – **Refinement:** The TSF shall restrict the ability to [selection: change default, query, modify, delete, clear, **[selection: [assignment: other operations], none]**] **all** the [TSF data except cryptographic security data and the time and date used to form the time stamps in FPT_STM.1] to [the administrators or authorized IT entities].

Application Note: The ST author should iterate this requirement as necessary to ensure that the TSF data are characterized in terms of the functionality provided by the TOE, and that the access is appropriately restricted to the appropriate administrators and authorized IT entities. The cryptographic security data and time stamp data are covered in the following two components, as they have specific requirements to address the PP's threats and policies.

FMT_MTD.1(2) Management of TSF data (cryptographic TSF data)

FMT_MTD.1.1(2) - The TSF shall restrict the ability to *modify* the [cryptographic security data] to [the Cryptographic Administrator].

Application Note: The intent of this requirement is to restrict the ability to configure the TOE's cryptographic policy to the Cryptographic Administrator. Configuring the cryptographic policy is related to things such as: setting modes of operation, key

lifetimes, selecting a specific algorithm, and key length.

FMT_MTD.1(3) Management of TSF data (time TSF data)

FMT_MTD.1.1(3) – The TSF shall restrict the ability to [*set*] the [time and date used to form the time stamps in FPT_STM.1] to [the Security Administrator or authorized IT entity].

Application Note: The ST author is able to restrict the ability to set the time and date to the just the Security Administrator, to just an authorized IT entity (e.g., NTP server) or both.

FMT_MTD.1(4) Management of TSF data (Information flow policy ruleset)

FMT_MTD.1.1(4) – The TSF shall restrict the ability to *query, modify, delete, create,* [selection: [assignment: other operations], none] the [information flow policy rules] to [the Security Administrator].

Application Note: This restricts the specification of the information flow policy ruleset identified in the FDP_IFF requirements to the Security Administrator. This specification is done using the attributes defined for those policies.

The ST author should fill in any TOE-specific operations that an administrator can perform on the ruleset in the assignment.

FMT_MTD.2(1) Management of limits on TSF data (transport-layer quotas)

FMT_MTD.2.1(1) - The TSF shall restrict the specification of the limits for [quotas on transport-layer connections] to [the Security Administrator].

FMT_MTD.2.2(1) - The TSF shall take the following actions, if the TSF data are at, or exceed, the indicated limits: [assignment: actions to be taken].

Application Note: Note that the wording of FRU_RSA.1(1) does not indicate that the TOE must provide the Security Administrator the means to adjust the maximum quota; however, if the TOE does provide such a mechanism then FMT_MTD.2.1(1) would require that that mechanism is restricted to the Security Administrator.

For FMT_MTD.2.2(1), the ST author should specify the actions that the TOE takes when quota is reached. For the TCP SYN attack, for example, the action might be to drop the oldest "n" half-open connections.

FMT_MTD.2(2) Management of limits on TSF data (controlled connection-oriented quotas)

FMT_MTD.2.1(2) - The TSF shall restrict the specification of the limits for [quotas on controlled connection-oriented resources] to [the Security Administrator].

FMT_MTD.2.2(2) - The TSF shall take the following actions, if the TSF data are at, or exceed, the indicated limits: [assignment: actions to be taken].

Application Note: For FMT_MTD.2.2(2), the ST author should specify the actions that the TOE takes for each controlled connection-oriented resource when the quota (with respect the specific network identifier or set of network identifiers) established by the Security Administrator is reached. This requirement may have to be iterated to be consistent with FRU_RSA.1(2). See the application note on FRU_RSA.1(2) for more detail on the requirements for the quota mechanism.

FMT_REV.1 Revocation

FMT_REV.1.1 – **Refinement:** The TSF shall restrict the ability to revoke security attributes associated with the **[users, information flow policy ruleset, services available to unauthenticated users, [selection: [assignment: other resources], none]]** within the TSC to [the Security Administrator].

Application Note: The security attributes associated with users are defined in FIA_ATD.1; the intent is to include an indication that a user is allowed to act in a role (Security Administrator Cryptographic Administrator or Audit Administrator).

The security attributes associated with the information flow policy ruleset are the rules themselves, and any attributes listed in the FDP_IFF.1.1-NIAP-0417 elements that are grouped to create new attributes that can be used in forming a rule.

The security attributes associated with the services available to unauthenticated users is just the list of services.

The ST author should specify all other resources that may have "revocable" aspects as implemented in the TOE, and ensure that FMT_REV.1.2 specifies rules for these resources. This list may be empty in an ST.

FMT_REV.1.2 - **Refinement:** The TSF shall **immediately** enforce the:

- [revocation of a user's role (Security Administrator, Cryptographic Administrator, Audit Administrator);
- changes to the information flow policy ruleset when applied;
- disabling of a service available to unauthenticated users; and
- [selection: [assignment: other rules], none]].

Application Note: The ST author should specify any rules covering additional resources detailed in the assignment in FMT_REV.1.1.

FMT_SMR.2 Restrictions on security roles

FMT_SMR.2.1 - The TSF shall maintain the roles:

- [Security Administrator;
- Cryptographic Administrator (i.e., users authorized to perform cryptographic initialization and management functions);
- Audit Administrator; and
- [selection: [assignment: any other roles], none]].

FMT_SMR.2.2 - The TSF shall be able to associate users with roles.

FMT_SMR.2.3 - The TSF shall ensure that the conditions

- [All roles shall be able to administer the TOE locally;
- all roles shall be able to administer the TOE remotely;
- all roles are distinct; that is, there shall be no overlap of operations performed by each role, with the following exceptions:
- all administrators can review the audit trail; and
- all administrators can invoke the self-tests] are satisfied.

Application Note: The administering of the TOE is limited to the capabilities associated with each administrative role. When the term administrator is used in this PP it refers to a person acting in any of the roles specified in FMT_SMR.2.1. The FIPS 140 validated cryptographic module for this TOE (level 3 for Roles) requires that unique trusted user identifiers be assigned to administer the cryptographic module. Only users associated with the Security Administrator role are allowed to administer the cryptographic module.

5.1.6 Protection of the TOE Security Functions (FPT)

FPT_RCV.1 Manual Recovery

FPT_RCV.1.1 - After a failure or service discontinuity, the TSF shall enter a maintenance mode where the ability to return the TOE to a secure state is provided.

FPT_RPL.1 Replay detection

FPT_RPL.1.1 - The TSF shall detect replay for the following entities: [TSF data and security attributes].

FPT_RPL.1.2 - The TSF shall perform

- [reject data;
- audit event; and
- [selection: [assignment: list of specific actions], none]]

when replay is detected.

Application Note: Receiving multiple network packets due to network congestion or lost packet acknowledgments is not considered a replay attack. The intent of this requirement is to ensure that an administrative session (in part, in its entirety, by a remote administrator or an authorized IT entity) or a user's authentication sequence cannot be replayed.

FPT_RVM.1 Non-bypassability of the TSP

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.

FPT_SEP.2 SFP domain separation

FPT_SEP.2.1 - The unisolated 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 the part of the TSF related to [**cryptography**] in **an address space** for its own execution that protects **it** from interference and tampering by the remainder of the TSF and by subjects untrusted with respect to **the cryptographic functionality**.

Application Note: The address space protection would be only for accidental interference (e.g., coding errors) but not from any malicious part of the kernel. It does protect against malicious untrusted subjects. Off board hardware or a third processor hardware state is a preferred implementation, because it would protect the cryptography from all other parts of the TSF. Cryptographic functionality is implemented in cryptomodules as well as by other code residing in the TSF that has not been validated through the FIPS 140-2 process. All cryptographic functionality, whether implemented in a cryptomodule or in some other way, is covered by the third element of this component.

FPT_STM.1 Reliable time stamps

FPT_STM.1.1 - The TSF shall be able to provide reliable time stamps for its own use.

FPT_TST_EXP.4 TSF testing (with cryptographic integrity verification)

FPT_TST_EXP.4.1 –The TSF shall run a suite of self-tests during initial start-up, periodically during normal operation as specified by the Security Administrator, and at the request of an administrator to demonstrate the correct operation of the hardware portions of the TSF.

FPT_TST_EXP.4.2 –The TSF shall provide an administrator with the capability to use a TSF-provided cryptographic function to verify the integrity of all TSF data except the following: audit data, [selection: [assignment: other dynamic TSF data for which no integrity validation is justified], none].

FPT_TST_EXP.4.3 - The TSF shall provide an administrator with the capability to use a TSF-provided cryptographic function to verify the integrity of stored TSF executable code.

FPT_TST_EXP.4.4 - The TSF shall restrict the ability to invoke the self-tests to an Administrator.

Application Note: This explicit requirement is necessary since some TOE data are dynamic (e.g., data in the audit trail, passwords) and so interpretation of "integrity" for FPT_TST.1.2 is required, leading to potential inconsistencies. The intention is that any parameter that only an administrator can control is verified to ensure its integrity is maintained. It is not necessary for the TOE to verify the integrity of audit data or user's passwords. If the TOE verifies the integrity of these, the ST author may fill in the assignment to include them.

Since this TOE includes all the hardware necessary for the operation of the TOE, the element FPT_TST_EXP.4.1 ensures that the hardware aspects of the TOE are tested prior to or during operations. It is not necessary to test the software portions of the TSF, since the evaluation ensures the correct operation of the software, software does not degrade or suffer intermittent faults, as does hardware, and integrity of the software portions of the TSF are addressed by FPT_TST_EXP.4.3. Note that since cryptographic functions implemented in hardware that are part of a cryptomodule are tested in FPT_TST_EXP.5, this requirement only applies to cryptographic functionality implemented in hardware that is not implemented in a cryptomodule (for instance, an implementation of a Key Agreement algorithm).

In element 4.2, the ST author should specify the TSF data for which integrity validation is not required, and also specify the administrative role that is able to invoke the integrity verification process. While some TSF data are dynamic and therefore not amenable to integrity verification, it is expected that all TSF data for which integrity verification "makes sense" be subject to this requirement.

In elements 4.2 and 4.3, the cryptographic mechanism can be any one of the ones specified in FCS_COP_EXP.3 or FCS_COP_EXP.6, although typically hash functions or digital signatures are used for integrity verification.

FPT_TST_EXP.5 Cryptographic self-test

FPT_TST_EXP.5.1 – The TSF shall run the suite of self-tests provided by the FIPS 140-2 cryptographic module during initial start-up (power on), at the request of an administrator, periodically (at a Security Administrator-specified interval not less than at least once a day) to demonstrate the correct operation of the cryptographic components of the TSF.

FPT_TST_EXP.5.2 – The TSF shall be able to run the suite of self-tests provided by the FIPS 140-2 cryptographic module immediately after the generation of a key.

FPT_TST_EXP.5.3 - The TSF shall restrict the ability to invoke these self-tests to an Administrator.

Application Note: For element 5.2, the Cryptographic Administrator has the ability to enable and disable this capability; this is specified in FMT_MOF.1(2). This requirement goes beyond what is required in FIPS140-2 for self-tests, in that the self-tests must be executable on demand rather than just at power-up.

5.1.7 Resource allocation (FRU_RSA)

FRU_RSA.1(1) - Maximum quotas (transport-layer quotas)

FRU_RSA.1.1(1) – **Refinement:** The TSF shall enforce maximum quotas of the following resources: [transport-layer representation] that **a *source subject identifier*** can use *over a specified period of time*.

Application Note: "transport-layer representation" refers specifically to the TCP SYN attack, where half-open connections are established thus exhausting the connection table resource. The selection for this requirement was refined to specify a source subject identifier, which is more accurate that user or subject in the context of a firewall. If the TOE does not implement the TCP/IP protocol, this requirement would apply to a similar type of transport-layer entity for that TOE's protocol stack.

FRU_RSA.1(2) - Maximum quotas (controlled connection-oriented quotas)

FRU_RSA.1.1(2) – **Refinement:** The TSF shall enforce **administrator-specified** maximum quotas of the following resources: [controlled connection-oriented resources] that ***users associated with an administrator-specified network identifier and a set of***

administrator-specified network identifiers can use *over* ***an administrator****-specified period of time*.

Application Note: This requirement applies to a network entity attempting to exhaust the specified connection-oriented resources (or set of such resources) on the TOE. Connectionless sessions are not a concern because they do not consume resources that persist like connection-oriented sessions do.

The ST author should fill in the first assignment with the list of connection-oriented resources to which this requirement applies. That is, when a network entity uses such a connection-oriented resource (or a collection of these resources), the TOE tracks that use for the purpose of determining whether the entity has exceed the quota established by the administrator.

The ST author should use the first selection to indicate whether the TOE is able to track the assignment of the specified resources based on a single network identifier (e.g., a specific IP address) or multiple network identifiers (e.g., a specific IP subnet address). The second selection should reflect the way in which the TOE tracks such resource use. Note that the ST author may have to iterate this requirement if different resources can be controlled differently by the TOE. The ST author should ensure that FMT_MTD.2(2) specifies the actions that are taken for each resource on which there is a quota.

5.1.8 TOE Access (FTA)

FTA_SSL.1 TSF-initiated session locking

FTA_SSL.1.1 – **Refinement:** The TSF shall lock a **local** interactive session after [a Security Administrator-specified time period of inactivity] by:

- clearing or overwriting display devices, making the current contents unreadable;
- 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** prior to unlocking the session.

FTA_SSL.2 User-initiated locking

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

- clearing or overwriting display devices, making the current contents unreadable;

- 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** prior to unlocking the session.

Application Note: The interactive sessions in FTA_SSL.1 and FTA_SSL.2 are those of the local administrator. Non-administrators only have remote access to the TOE and the requirements for session locking levied on them are specified in FTA_SSL.3.

FTA_SSL.3 TSF-initiated termination

FTA_SSL.3.1 - **Refinement:** The TSF shall terminate **a remote** session after a [Security Administrator-configurable time interval of session inactivity].

Application Note: A remote session applies to remote administrators and any connection to a service on the firewall as defined in FDP_IFF.1.3-NIAP-0417(1)(a).

FTA_TAB.1 Default TOE access banners

FTA_TAB.1.1 - **Refinement:** Before establishing a user session **that requires authentication**, the TSF shall display **only a Security Administrator-specified** advisory **notice and consent** warning message regarding unauthorized use of the TOE.

Application Note: The access banner applies whenever the TOE will provide a prompt for identification and authentication (e.g., administrators). The intent of this requirement is to advise users of warnings regarding the unauthorized use of the TOE and to provide the Security Administrator with control over what is displayed (e.g., if the Security Administrator chooses, they can remove banner information that informs the user of the product and version number).

FTA_TSE.1 TOE session establishment

FTA_TSE.1.1 - **Refinement:** The TSF shall be able to deny establishment **of an authorized user session** based on [location, time, and day].

5.1.9 Trusted Path/Channels (FTP)

FTP_ITC.1(1) Inter-TSF trusted channel (Prevention of Disclosure)

FTP_ITC.1.1(1) - **Refinement:** The TSF shall **use encryption to** provide a **trusted** communication channel between itself and **authorized IT entities** that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure.

FTP_ITC.1.2(1) **Refinement:** The TSF shall permit *the TSF*, ***or the authorized IT entities*** to initiate communication via the trusted channel.

Application Note: The encryption used to protect the communication channel from disclosure is the symmetric algorithm specified in FCS_COP_EXP.2.

FTP_ITC.1.2 is used to ensure secure communications between the TOE and authorized IT entities (e.g., certificate authority server). While these authorized IT entities may initiate communications, it may be the case that the TOE is required to perform a "pull" operation (e.g., obtaining a certificate from a certificate authority, obtaining time from an NTP server).

FTP_ITC.1.3(1) - The TSF shall initiate communication via the trusted channel for [all authentication functions, [selection: [assignment: list of other functions for which a trusted channel is required], none]].

Application Note: The "other functions" are the services that are provided by the authorized IT entities (e.g., NTP).

FTP_ITC.1(2) Inter-TSF trusted channel (Detection of Modification)

FTP_ITC.1.1(2) - **Refinement:** The TSF shall **use a cryptographic signature** to provide a **trusted** communication channel between itself and **authorized IT entities** that is logically distinct from other communication channels and provides assured identification of its end points and **detection of the modification of data**.

FTP_ITC.1.2(2) - **Refinement:** The TSF shall permit *the TSF*, ***or the authorized IT entities*** to initiate communication via the trusted channel.

Application Note: The method used to provide detection of data modification transmitted through the communication channel is the cryptographic digital signature algorithm specified in FCS_COP_EXP.3.

FTP_ITC.1.2 is used to ensure secure communications between the TOE and authorized IT entities (e.g., certificate authority server). While these authorized IT entities may initiate communications, it may be the case that the TOE is required to perform a "pull" operation (e.g., obtaining a certificate from a certificate authority, obtaining time from an NTP server).

FTP_ITC.1.3(2) - The TSF shall initiate communication via the trusted channel for [all authentication functions, [selection: [assignment: list of other functions for which a trusted channel is required], none]].

Application Note: The "other functions" are the services that are provided by the authorized IT entities (e.g., NTP).

FTP_TRP.1(1) Trusted path (Prevention of Disclosure)

FTP_TRP.1.1(1) - **Refinement:** The TSF shall provide **an encrypted** communication path between itself and *remote* ***administrators and authenticated proxy*** users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from disclosure.

FTP_TRP.1.2(1) - The TSF shall permit *remote users* to initiate communication via the trusted path.

FTP_TRP.1.3(1) – **Refinement:** The TSF shall require the use of the trusted path for ***user authentication, all remote administration actions, [selection: [assignment: other services for which trusted path is required, none]].***

Application Note: The encryption used to protect the communication channel from disclosure is the symmetric algorithm specified in FCS_COP_EXP.2

"all remote administration actions" means that the entire remote administration session is protected with the trusted path; that is, the administrator is assured of communicating with the TOE and the data passing between the administrator and the TOE are protected from disclosure.

FTP_TRP.1(2) Trusted path (Detection of Modification)

FTP_TRP.1.1(2) - **Refinement:** The TSF shall **use a cryptographic signature to** provide a communication path between itself and *remote* ***administrators and authenticated proxy*** users that is logically distinct from other communication paths and provides assured identification of its end points and **detection of the** modification of data.

FTP_TRP.1.2(2) - The TSF shall permit *remote users* to initiate communication via the trusted path.

FTP_TRP.1.3(2) – **Refinement:** The TSF shall require the use of the trusted path for ***user authentication, all remote administration actions, [selection: [assignment: other services for which trusted path is required, none]]***.

Application Note: The method used to provide detection of data modification transmitted through the communication channel is the cryptographic digital signature algorithm specified in FCS_COP_EXP.3.

"all remote administration actions" means that the entire remote administration session is protected with the trusted path; that is, the administrator is assured of communicating with the TOE and the data passing between the administrator and the TOE provides a means for detecting the modification of data that flows through the protected communication path.

5.2 Security Requirements for the IT Environment

This Protection Profile provides functional requirements for the IT Environment. The

IT environment includes authorized IT entities (e.g., a certificate authority server, NTP server) and any IT entities that are used by administrators to remotely administer the TOE. These requirements consist of functional components from Part 2 of the CC.

FTP_ITC.1(1) Inter-TSF trusted channel (Prevention of Disclosure)

FTP_ITC.1.1(1) - **Refinement:** The **IT Environment** shall provide a **trusted** communication channel between itself and the **TSF** that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure.

FTP_ITC.1.2(1) - **Refinement:** The TSF shall permit *the TSF*, or ***the IT Environment*** to initiate communication via the trusted channel.

FTP_ITC.1.3(1) - The TSF shall initiate communication via the trusted channel for [all authentication functions, [selection: [assignment: communications with authorized IT entities], none]].

Application Note: If a certificate authority server plays a role in the authentication of users, then the CA is considered an authorized IT entity and the TSF is expected to initiate secure communications with this entity. If the TSF makes use of an NTP server, it is expected that the TSF would initiate the trusted channel with the NTP server.

FTP_ITC.1(2) Inter-TSF trusted channel (Detection of Modification)

FTP_ITC.1.1(2) - **Refinement:** The **IT Environment** shall provide **an encrypted** communication channel between itself and **the TSF** that is logically distinct from other communication channels and provides assured identification of its end points and **detection of the modification of data.**

FTP_ITC.1.2(2) - **Refinement:** The TSF shall permit *the TSF*, or ***the IT Environment*** to initiate communication via the trusted channel.

FTP_ITC.1.3(2) - The TSF shall initiate communication via the trusted channel for [all authentication functions, [selection: [assignment: communications with authorized IT entities], none]].

Application Note: If a certificate authority server plays a role in the authentication of users, then the CA is considered an authorized IT entity and the TSF is expected to initiate secure communications with this entity. If the TSF makes use of an NTP server, it is expected that the TSF would initiate the trusted channel with the NTP server.

FTP_TRP.1(1) Trusted path (Prevention of Disclosure)

FTP_TRP.1.1(1) - **Refinement:** The **IT Environment** shall provide an **encrypted** communication path between itself and **the TSF** that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from disclosure.

FTP_TRP.1.2(1) - The **IT Environment** shall permit *remote users* ***of the TSF*** to initiate communication to the TSF via the trusted path.

FTP_TRP.1.3(1) – **Refinement:** The **IT Environment** shall **initiate** the use of the trusted path for *user authentication*, ***all remote administration actions***, ***[selection: [assignment: other services for which trusted path is required, none]]***.

Application Note: The encryption used to protect the communication channel from disclosure is the symmetric algorithm specified in FCS_COP_EXP.2.

This requirement (as is FTP_ITC.1) is levied on the IT environment to ensure that the necessary support exists in the IT environment to communicate securely with the TOE. The FCS family of requirements has not been explicitly stated in the IT environment requirements, since the cryptographic algorithms and key sizes are implicitly required by the IT environment in order to communicate with the TOE.

FTP_TRP.1(2) Trusted path (Detection of Modification)

FTP_TRP.1.1(2) - **Refinement:** The **IT Environment** shall provide an **encrypted** communication path between itself and **the TSF** that is logically distinct from other communication paths and provides assured identification of its end points **and detection of the modification of data.**

FTP_TRP.1.2(2) - The **IT Environment** shall permit *remote users* ***of the TSF*** to initiate communication **to the TSF** via the trusted path.

FTP_TRP.1.3(2) – **Refinement:** The **IT Environment** shall **initiate** the use of the trusted path for *user authentication*, ***all remote administration actions***, ***[selection: [assignment: other services for which trusted path is required, none]]***.

Application Note: The method used to provide detection of data modification transmitted through the communication channel cryptographic signature algorithm specified in FCS_COP_EXP.3.

This requirement (as is FTP_ITC.1) is levied on the IT environment to ensure that the necessary support exists in the IT environment to communicate securely with the TOE. The FCS family of requirements has not been explicitly stated in the IT environment requirements, since the cryptographic algorithms and key sizes are implicitly required by the IT environment in order to communicate with the TOE.

5.3 TOE Security Assurance Requirements

The TOE assurance requirements for this PP do not map to a CC EAL. The assurance requirements are summarized in the Table 4 below, with the explicit requirements in bold print. The objectives and application notes for the explicit ADV requirements are contained in Section 7. The methodology for performing the evaluation activities pertaining to the explicit assurance requirements is provided by CCEVS management in a separate document.

Assurance Class	Assurance Components	
Configuration management	ACM_AUT.1	Partial CM automation
	ACM_CAP.4	Generation support and acceptance procedures
	ACM_SCP.2	Problem tracking CM coverage
Delivery and operation	ADO_DEL.2	Detection of modification
	ADO_IGS.1	Installation, generation, and start-up procedures
Development	**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_INT_EXP.1	**Modular Decomposition**
	ADV_IMP.2	Implementation of the TSF
	ADV_LLD_EXP.1	**Security-Enforcing Low-Level design**
	ADV_RCR.1	Informal correspondence demonstration
	ADV_SPM.1	Informal TOE security policy model
Guidance documents	AGD_ADM.1	Administrator guidance
	AGD_USR.1	User guidance
Life cycle support	ALC_DVS.1	Identification of security measures
	ALC_FLR.2	Flaw Reporting Procedures
	ALC_LCD.1	Developer defined life-cycle model

Assurance Class	Assurance Components	
	ALC_TAT.1	Well-defined development tools
Tests	ATE_COV.2	Analysis of coverage
	ATE_DPT.2	Testing: low-level design
	ATE_FUN.1	Functional testing
	ATE_IND.2	Independent testing - sample
Vulnerability assessment	**AVA_CCA_EXP.2**	**Systematic cryptographic module covert channel analysis**
	AVA_MSU.2	Validation of analysis
	AVA_SOF.1	Strength of TOE security function evaluation
	AVA_VLA.3	Moderately resistance

Table 4 – Assurance Requirements

ACM_AUT.1 Partial CM automation

Developer action elements:

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

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

Content and presentation of evidence elements:

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.

Evaluator action elements:

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

ACM_CAP.4 Generation support and acceptance procedures

Developer action elements:

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.

Content and presentation of evidence elements:

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.

ACM_CAP.4.7C - The CM system shall uniquely identify all configuration items.

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.

Evaluator action elements:

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

ACM_SCP.2 Problem tracking CM coverage

Developer action elements:

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

Content and presentation of evidence elements:

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

Evaluator action elements:

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

ADO_DEL.2 Detection of modification

Developer action elements:

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.

Content and presentation of evidence elements:

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.

Evaluator action elements:

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

ADO_IGS.1 Installation, generation, and start-up procedures

Developer action elements:

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

Content and presentation of evidence elements:

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

Evaluator action elements:

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.

ADV_ARC_EXP.1 Architectural Description

Developer action elements:

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

Content and presentation of evidence elements:

ADV_ARC_EXP.1.1C The presentation of the architectural design of the TSF shall be informal.

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.

Evaluator action elements:

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.

ADV_FSP_EXP.1 Functional Specification with Complete Summary

Developer action elements:

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

Content and presentation of evidence elements:

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) using an informal style.

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.

Evaluator action elements:

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.

> *Application Note: This requirement can potentially be met by a combination of documents provided by the developer, including the Security Target and external interface specification.*

ADV_HLD_EXP.1 Security-Enforcing High-Level design

Developer action elements:

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

Content and presentation of evidence elements:

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 subsystems using an informal style.

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 other 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.

Evaluator action elements:

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.

ADV_INT_EXP.1 Modular Decomposition

Developer action elements:

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 the modules such that they exhibit good internal structure and are not overly complex.

ADV_INT_EXP.1.4D The developer shall design modules that implement the [FDP_IFC.1(1), FDP_IFC.1(2) FDP_IFF.1(1),and FDP_IFF.1(2), requirements] 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 modules such that they exhibit only call or common coupling, with limited exceptions.

ADV_INT_EXP.1.6D The developer shall implement TSF 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.

Content and presentation of evidence elements:

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 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.

Evaluator action elements:

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.

ADV_IMP.2 Implementation of the TSF

Developer action elements:

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

Content and presentation of evidence elements:

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

ADV_IMP.2.2C The implementation representation shall be internally consistent.

ADV_IMP.2.3C The implementation representation shall describe the relationships between all portions of the implementation.

Evaluator action elements:

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

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

ADV_LLD_EXP.1 Security-Enforcing Low-Level Design

Developer action elements:

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

Content and presentation of evidence elements:

ADV_LLD_EXP.1.1C The presentation of the low-level design shall be informal.

ADV_LLD_EXP.1.2C The presentation of the low-level design shall be separate from the implementation representation.

ADV_LLD_EXP.1.3C The low-level design shall be internally consistent.

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 the TSF in terms of modules, designating each module as either security-enforcing or security-supporting.

ADV_LLD_EXP.1.6C The low level design shall describe each security-enforcing module in terms

of its purpose, interfaces, return values from those interfaces, called interfaces to other modules, and global variables.

ADV_LLD_EXP.1.7C 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.1.8C The low level design shall describe each security-supporting module in terms of its purpose and interaction with other modules.

Evaluator action elements:

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.

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

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

Content and presentation of evidence elements:

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.

Evaluator action elements:

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

> *Application Note: The intent of this requirement is for the vendor to provide, and the evaluator to confirm, that there exists accurate, consistent, and clear mappings between each level of design decomposition. Thus there can be no TOE security functions defined at a lower layer of abstraction absent from a higher level of abstraction and vice versa.*

ADV_SPM.1 Informal TOE security policy model

Developer action elements:

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.

Content and presentation of evidence elements:

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.

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.

> *Application Note: As part of the secure state, the cryptographic module is in a known state such that all critical areas are empty of plaintext/red/secret data and inaccessible to processes, and all security policies are enforced.*

Evaluator action elements:

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

AGD_ADM.1 Administrator guidance

Developer action elements:

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

Content and presentation of evidence elements:

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

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.

Evaluator action elements:

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

AGD_USR.1 User guidance

Developer action elements:

AGD_USR.1.1D - The developer shall provide user guidance.

Content and presentation of evidence elements:

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

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.

Evaluator action elements:

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

ALC_DVS.1 Identification of security measures

Developer action elements:

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

Content and presentation of evidence elements:

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.

Evaluator action elements:

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.

ALC_FLR.2 Flaw Reporting Procedures

ALC_FLR.2.1D - The developer shall document the flaw remediation procedures.

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.

Content and presentation of evidence elements:

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

ALC_FLR.2.6C - The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.

Evaluator action elements:

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

ALC_LCD.1 Developer defined life-cycle model

Developer action elements:

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.

Content and presentation of evidence elements:

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.

Evaluator action elements:

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

ALC_TAT.1 Well-defined development tools

Developer action elements:

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.

Content and presentation of evidence elements:

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

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.

Evaluator action elements:

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

ATE_COV.2 Analysis of coverage

Developer action elements:

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

Content and presentation of evidence elements:

ATE_COV.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.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.

Evaluator action elements:

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

ATE_DPT.2 Testing: low-level design

Developer action elements:

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

Content and presentation of evidence elements:

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.

Evaluator action elements:

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

ATE_FUN.1 Functional testing

Developer action elements:

ATE_FUN.1.1D - The developer shall test the TSF and document the results.

ATE_FUN.1.2D - The developer shall provide test documentation.

Content and presentation of evidence elements:

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.

Evaluator action elements:

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

ATE_IND.2 Independent testing - sample

Developer action elements:

ATE_IND.2.1D - The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

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.

Evaluator action elements:

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.

AVA_CCA_EXP.2 Systematic Cryptographic Module Covert Channel Analysis

Application Note: The covert channel analysis is performed on the entire TSF to determine that TSF interfaces cannot be used covertly to obtain critical security parameters; a search is made for the leakage of critical security parameters, rather than a violation of an information control policy.

Developer action elements:

AVA_CCA_EXP.2.1D - The developer shall conduct a search for covert channels for the leakage of critical security parameters.

AVA_CCA_EXP.2.2D - The developer shall provide covert channel analysis documentation.

Content and presentation of evidence elements:

AVA_CCA_EXP.2.1C - The analysis documentation shall identify covert channels that leak critical security parameters and estimate their capacity.

AVA_CCA_EXP.2.2C - The analysis documentation shall describe the procedures used for determining the existence of covert channels that leak critical security parameters, 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.

Evaluator action elements:

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

AVA_CCA_EXP.2.2E - The evaluator shall selectively validate the covert channel analysis through independent analysis and testing.

Application Note: The cryptographic security parameters are defined in FIPS 140-2.

AVA_MSU.2 Validation of analysis

Developer action elements:

AVA_MSU.2.1D - The developer shall provide guidance documentation.

AVA_MSU.2.2D - The developer shall document an analysis of the guidance documentation.

Content and presentation of evidence elements:

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.

Evaluator action elements:

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.

AVA_SOF.1 Strength of TOE security function evaluation

Developer action elements:

AVA_SOF.1.1D - The developer shall perform a strength of TOE security function analysis for each mechanism identified in the Security Target as having a strength of TOE security function claim.

Content and presentation of evidence elements:

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.

Evaluator action elements:

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.

AVA_VLA.3 Moderately resistant

Developer action elements:

AVA_VLA.3.1D - The developer shall perform a vulnerability analysis.

AVA_VLA.3.2D - The developer shall provide vulnerability analysis documentation.

Content and presentation of evidence elements:

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.

Evaluator action elements:

AVA_VLA.3.1E - The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.3.2E - The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure the identified vulnerabilities have been addressed.

AVA_VLA.3.3E - The evaluator shall perform an independent vulnerability analysis.

AVA_VLA.3.4E - The 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 - The evaluator shall determine that the TOE is resistant to penetration attacks performed by an attacker possessing a moderate attack potential.

6.0 RATIONALE

This section describes the rationale for the Security Objectives and Security Functional Requirements as defined in Section 4 and Section 5, respectively. Additionally, this section describes the rationale for not satisfying all of the dependencies and the rationale for the strength of function (SOF) claim. Table 5 illustrates the mapping from Security Objectives to Threats and Policies.

6.1 Rationale for TOE Security Objectives

Threat/Policy	Objectives Addressing the Threat	Rationale
T.ADDRESS_MASQUERADE A user on one interface may masquerade as a user on another interface to circumvent the TOE policy.	O.MEDIATE The TOE must mediate the flow of information between sets of TOE network interfaces or between a network interface and the TOE itself in accordance with its security policy.	O.MEDIATE (FDP_IFC.1, FDP_IFF.1-NIAP-0417(1), (FDP_IFC.1(2), FDP_IFF.1-NIAP-0417(2),) counters this threat by ensuring that all network packets that flow through the TOE are subject to the information flow policies. The rules in each of the policies ensure that the network identifier in a network packet is in the set of network identifiers associated with a TOE's network interface. Therefore, if a user supplied a network identifier in a packet that was associated with a TOE network interface other than the one the user supplied the packet on, the packet would not be allowed to flow through the TOE, or access TOE services. This would, for example, prevent a user from sending a packet from the Internet claiming to be on a machine on the protected enclave.
T.ADMIN_ERROR An administrator may incorrectly install or configure the TOE, or install a corrupted TOE resulting in ineffective security mechanisms.	O.ROBUST_ADMIN_GUIDANCE The TOE will provide administrators with the necessary information for secure delivery and management. O.ADMIN_ROLE The TOE will provide an administrator role to isolate administrative actions. O.MANAGE The TOE will provide all the functions and facilities necessary to support the administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.	O.ROBUST_ADMIN_GUIDANCE (ADO_DEL.2, ADO_IGS.1, AGD_ADM.1, AGD_USR.1, AVA_MSU.2) help to mitigate this threat by ensuring the TOE administrators have guidance that instructs them how to administer the TOE in a secure manner and to provide the administrator with instructions to ensure the TOE was not corrupted during the delivery process. Having this guidance helps to reduce the mistakes that an administrator might make that could cause the TOE to be configured in a way that is unsecure. O.ADMIN_ROLE (FMT_SMR.2) plays a role in mitigating this threat by limiting the functions an administrator can perform in a given role. So for example, the Audit Administrator could not make a configuration mistake that would impact the TOE's ability to mediate information flow. O.MANAGE (FMT_MTD.1(1), FMT_MTD.1(4)) also contributes to mitigating this threat by providing administrators the capability to view configuration settings. For example, if the Security Administrator made a mistake when configuring the ruleset, providing them the capability to view the rules affords them the ability to review the rules and discover any mistakes that might have been made.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.ADMIN_ROGUE An administrator's intentions may become malicious resulting in user or TSF data being compromised.	O.ADMIN_ROLE The TOE will provide an administrator role to isolate administrative actions.	O.ADMIN_ROLE (FMT_SMR.2) mitigates this threat to a limited degree by limiting the functions available to an administrator. This is somewhat different than the part this objective plays in countering T.ADMIN_ERROR, in that this presumes that separate individuals will be assigned separate roles. If the Audit Administrator's intentions become malicious they would not be able to render the TOE unable to enforce its information flow policies. On the other hand, if the Security Administrator becomes malicious they could affect the information flow policy, but the Audit Administrator may be able to detect those actions.
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.	O.AUDIT_PROTECTION The TOE will provide the capability to protect audit information. O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated. O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure.	O.AUDIT_PROTECT (FAU.SAR.2, FAU_STG.1-NIAP-0423, FAU_STG.3, FAU_STG.NIAP-0414-1-NIAP-0429, FMT_MOF.1(2)) contributes to mitigating this threat by controlling access to the audit trail. No one is allowed to modify audit records, the Audit Administrator is the only one allowed to delete the audit trail. The TOE has the capability to prevent auditable actions from occurring if the audit trail is full. O.RESIDUAL_INFORMATION (FDP.RIP.2) prevents a user not authorized to read the audit trail from access to audit information that might otherwise be persistent in a TOE resource (e.g., memory). By ensuring the TOE prevents residual information in a resource, audit information will not become available to any user or process except those explicitly authorized for that data. O.SELF_PROTECTION (FPT_SEP.2, FPT_RVM.1) contributes to countering this threat by ensuring that the TSF can protect itself from users. If the TSF could not maintain and control its domain of execution, it could not be trusted to control access to the resources under its control, which includes the audit trail.

Threat/Policy	Objectives Addressing the Threat	Rationale
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 compromise the cryptographic mechanisms and the data protected by those mechanisms.	O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated. O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure. O.DOCUMENT_KEY_LEAKAGE The bandwidth of channels that can be used to compromise key material shall be documented.	O.RESIDUAL_INFORMATION (FCS_CKM.4) mitigates the possibility of malicious users or processes from gaining inappropriate access to cryptographic data, including keys. This objective ensures that the cryptographic data does not reside in a resource that has been used by the cryptographic module and then reallocated to another process. O.SELF_PROTECTION (FPT_SEP.2, FPT_RVM.1) contributes to countering this threat by ensuring that the TSF can protect itself from users. If the TSF could not maintain and control its domain of execution, it could not be trusted to control access to the resources under its control, which includes the cryptographic data and executable code. O.DOCUMENT_KEY_LEAKAGE (AVA_CCA_EXP.2) addresses this threat by requiring the developer to perform an analysis that documents the amount of key information that can be leaked via a covert channel. This provides information that identifies how much material could be inappropriately obtained within a specified time period.
T.MASQUERADE A user may masquerade as an authorized user or an authorized IT entity to gain access to data or TOE resources.	O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate O.TRUSTED_PATH The TOE will provide a means to ensure users are not communicating with some other entity pretending to be the TOE, and that the TOE is communicating with an authorized IT entity and not some other entity pretending to be an authorized IT entity.	O.ROBUST_TOE_ACCESS (FIA_AFL.1-NIAP-0425, FIA_ATD.1, FIA_UID.2, FIA_UAU.1,FIA_UAU_EXP.5, FTA_TSE.1, AVA_SOF.1) mitigates this threat by controlling the logical access to the TOE and its resources. By constraining how and when authorized users can access the TOE, and by mandating the type and strength of the authentication mechanism this objective helps mitigate the possibility of a user attempting to login and masquerade as an authorized user. In addition, this objective provides the administrator the means to control the number of failed login attempts a user can generate before an account is locked out, further reducing the possibility of a user gaining unauthorized access to the TOE. O.TRUSTED_PATH (FTP_ITC.1(1), FTP_ITC.1(2)) ensures that the communication path end points between the TOE and authorized users (remote administrators, authorized IT entities) are defined. This mechanism allows the TOE to be assured that it is communicating with an authorized user. This also ensures that the transmitted data cannot be disclosed (e.g., encrypted). The protection offered by this objective is limited to TSF data and security attributes (e.g., proxy user's user data is not protected, since their session communication does not require encryption or any other form of protection).

Threat/Policy	Objectives Addressing the Threat	Rationale
T.FLAWED_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 design of the TOE will be the result of sound design principles and techniques; the design of the TOE, as well as the design principles and techniques, are adequately and accurately documented. O.VULNERABILITY_ANALYSIS_ TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	O.SOUND_DESIGN (ADV_FSP_EXP.1, ADV_HLD_EXP.1, ADV_INT_EXP.1, ADV_LLD_EXP.1, ADV_RCR.1, ADV_SPM.1) counters this threat, to a degree, by requiring that the TOE be developed using sound engineering principles. By accurately and completely documenting the design of the security mechanisms in the TOE, including a security model, the design of the TOE can be better understood, which increases the chances that design errors will be discovered. O.CHANGE_MANAGEMENT (ACM_AUT.1, ACM_CAP.4, ACM_SCP.2, ALC_DVS.1, ALC_FLR.2, ALC_LCD.1) plays a role in countering this threat by requiring the developer to provide control of the changes made to the TOE's design. This includes controlling physical access to the TOE's development area, and having an automated configuration management system that ensures changes made to the TOE go through an approval process and only those persons that are authorized can make changes to the TOE's design and its documentation. O.VULNERABILITY_ANALYSIS_TEST (AVA_VLA.3) ensures that the design of the TOE is independently analyzed for design flaws. Having an independent party perform the assessment ensures an objective approach is taken and may find errors in the design that would be left undiscovered by developers that have a preconceived incorrect understanding of the TOE's design.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.FLAWED_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.SOUND_IMPLEMENTATION The implementation of the TOE will be an accurate instantiation of its design, and is adequately and accurately documented. O.THOROUGH_FUNCTIONAL_ TESTING The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements. O.VULNERABILITY_ANALYSIS_ TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	O.CHANGE_MANAGEMENT (ACM_CAP.4, ACM_SCP.2, ALC_DVS.1, ALC_FLR.2, ALC_LCD.1, ACM_AUT.1) This objective plays a role in mitigating this threat in the same way that the flawed design threat is mitigated. By controlling who has access to the TOE's implementation representation and ensuring that changes to the implementation are analyzed and made in a controlled manner, the threat of intentional or unintentional errors being introduced into the implementation are reduced. In addition to documenting the design so that implementers have a thorough understanding of the design, O.SOUND_IMPLEMENTATION (ADV_IMP.2, ADV_LLD_EXP.1, ADV_RCR.1, ADV_INT_EXP.1, ALC_TAT.1) requires that the developer's tools and techniques for implementing the design are documented. Having accurate and complete documentation, and having the appropriate tools and procedures in the development process helps reduce the likelihood of unintentional errors being introduced into the implementation. Although the previous three objectives help minimize the introduction of errors into the implementation, O.THOROUGH_FUNCTIONAL_ TESTING (ATE_COV.2, ATE_FUN.1, ATE_DPT.2, ATE_IND.2) increases the likelihood that any errors that do exist in the implementation (with respect to the functional specification, high level, and low-level design) will be discovered through testing. O.VULNERABILITY_ANALYSIS_TEST (AVA_VLA.3) helps reduce errors in the implementation that may not be discovered during functional testing. Ambiguous design documentation, and the fact that exhaustive testing of the external interfaces is not required may leave bugs in the implementation undiscovered in functional testing. Having an independent party perform a vulnerability analysis and conduct testing outside the scope of functional testing increases the likelihood of finding errors.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.POOR_TEST Lack of or insufficient tests to demonstrate that all TOE security functions operate correctly (including in a fielded TOE) may result in incorrect TOE behavior being undiscovered.	O.CORRECT_ TSF_OPERATION The TOE will provide the capability to test the TSF to ensure the correct operation of the TSF in its operational environment. O.THOROUGH_FUNCTIONAL_ TESTING The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements. O.VULNERABILITY_ANALYSIS_ TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	Design analysis determines that TOE's documented design satisfies the 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.THOROUGH_FUNCTIONAL_ TESTING (ATE_FUN.1, ATE_COV.2, ATE_DPT.2, ATE_IND.2) ensures that adequate functional testing is performed to ensure the TSF satisfies the security functional requirements and demonstrates that 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.VULNERABILITY_ANALYSIS_TEST (AVA_VLA.3) 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 end users to ensure the TOE's security mechanisms continue to operate correctly once the TOE is fielded O.CORRECT_ TSF_OPERATION (FPT_TST_EXP.4, FPT_TST_EXP.5) 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 providing end users the confidence that the TOE's security policies continue to be enforced.
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 (captured as it was transmitted during the course of legitimate use).	O.REPLAY_DETECTION The TOE will provide a means to detect and reject the replay of TSF data and security attributes.	O.REPLAY_DETECTION (FPT_RPL.1) prevents a user from replaying TSF data and security attributes (e.g., TSF data or security attributes transmitted between a remote administrator, an authorized IT entity and the TOE) that could leave the TOE in a configuration that the administrative staff did not intend (e.g., an administrator modifies the auditable events to be recorded and a user captures that traffic. At a later date the administrator determines that the new set of auditable events is not sufficient and again modifies the events to be audited. The user then replays the earlier audit event configuration.).

Threat/Policy	Objectives Addressing the Threat	Rationale
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 information contained in a protected resource is not released when the resource is reallocated.	O.RESIDUAL_INFORMATION (FDP_RIP.2, FCS_CKM.4) counters this threat by ensuring that TSF data and user data is not persistent when resources are released by one user/process and allocated to another user/process. This means that network packets will not have residual data from another packet due to the padding of a packet.
T.RESOURCE_EXHAUSTION A malicious process or user may block others from system resources (e.g., connection state tables) via a resource exhaustion denial of service attack.	O.RESOURCE_SHARING The TOE shall provide mechanisms that mitigate attempts to exhaust connection-oriented resources provided by the TOE (e.g., entries in a connection state table; TCP connections used by proxies).	O.RESOURCE_SHARING (FRU_RSA.1(1), FRU_RSA.1(2), FMT_MTD.2(1), FMT_MTD.2(2), FMT_MOF.1(7)) mitigates this threat by requiring the TOE to provide controls over connection-oriented resources. These controls provide the administrator ability to specify which network identifiers have access to the TOE's connection-oriented resources over a time period that is specified by the administrator. This objective also addresses the denial-of-service attack of a user attempting to exhaust the connection-oriented resources by generating a large number of half-open connections (e.g., SYN attack).
T.SPOOFING An entity may mis-represent itself as the TOE to obtain authentication data.	O.TRUSTED_PATH The TOE will provide a means to ensure users are not communicating with some other entity pretending to be the TOE, and that the TOE is communicating with an authorized IT entity and not some other entity pretending to be an authorized IT entity.	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 (FTP_ITC.1(1), FTP_ITC.1(2), FTP_TRP.1(1), FTP_TRP.1(2)) 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.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.MALICIOUS_TSF_ COMPROMISE A malicious user or process may cause TSF data or executable code to be inappropriately accessed (viewed, modified, or deleted).	O.DISPLAY_BANNER The TOE will display an advisory warning regarding use of the TOE. O.MANAGE The TOE will provide all the functions and facilities necessary to support the administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use. O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated. O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure. O.TRUSTED_PATH The TOE will provide a means to ensure users are not communicating with some other entity pretending to be the TOE, and that the TOE is communicating with an authorized IT entity and not some other entity pretending to be an authorized IT entity.	O.TRUSTED_PATH (FTP_TRP.1(1), FTP_TRP.1(2), FTP_ITC.1(1), FTP_ITC.1(2)) plays a role in addressing this threat by ensuring that a trusted communication path exists between the TOE and authorized users (i.e., remote administrators, authorized IT entities). This ensures the transmitted data cannot be compromised or disclosed (e.g., encrypted) during the duration of the trusted path. The protection offered by this objective is limited to TSF data and security attributes (e.g., proxy user's user data is not protected, since their entire session communication (only the authentication portion of the session is protected) does not require encryption or any other form of protection). O.MANAGE (FMT_MTD.1(1)-(4), FMT_MSA.1, FMT_MOF.1(1)-(3)) is necessary because an access control policy is not specified to control access to TSF data. This objective is used to dictate who is able to view and modify TSF data, as well as the behavior of TSF functions. O.RESIDUAL_INFORMATION (FDP_RIP.2, FCS_CKM.4) is necessary to mitigate this threat, because even if the security mechanisms do not allow a user to explicitly view TSF data, if TSF data were to inappropriately reside in a resource that was made available to a user, that user would be able to inappropriately view the TSF data. O.SELF_PROTECTION (FPT_SEP.2, FPT_RVM.1) requires that the TSF be able to protect itself from tampering and that the security mechanisms in the TSF cannot be bypassed. Without this objective, there could be no assurance that users could not view or modify TSF data or TSF executables. O.DISPLAY_BANNER (FTA_TAB.1) helps mitigate this threat by providing the Security Administrator the ability to remove product information (e.g., product name, version number) from a banner that is displayed to users. Having product information about the TOE provides an attacker with information that may increase their ability to compromise the TOE.
T.UNATTENDED_SESSION A user may gain unauthorized access to an unattended session.	O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate	O.ROBUST_TOE_ACCESS (FTA_SSL.1, FTA_SSL.2, FTA_SSL.3) helps to mitigate this threat by including mechanisms that place controls on user's sessions. Local administrator's sessions are locked and remote sessions are dropped after a Security Administrator defined time period of inactivity. Locking the local administrator's session reduces the opportunity of someone gaining unauthorized access the session when the console is unattended. Dropping the connection of a remote session (after the specified time period) reduces the risk of someone accessing the remote machine where the session was established, thus gaining unauthorized access to the session.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.UNAUTHORIZED_ACCESS A user may gain access to services (by sending data through the TOE) for which they are not authorized according to the TOE security policy.	O.MEDIATE The TOE must mediate the flow of information between sets of TOE network interfaces or between a network interface and the TOE itself in accordance with its security policy.	O.MEDIATE (FDP_IFF.1-NIAP-0417(1), FDP_IFF.1-NIAP-0417(2), FDP_IFC.1(1), FDP_IFC.1(2), , FMT_REV.1, FPT_RVM.1) works to mitigate this threat by ensuring that all network packets that flow through the TOE are subject to the information flow policies. One of the rules ensures that the network identifier in a packet is in the set of network identifiers associated with a TOE's network interface. Therefore, if a user supplied a network identifier in a packet that purported to originate from a network associated with a TOE network interface other than the one the user supplied the packet on, the packet would not be allowed to flow through the TOE, or access TOE services. Another rule provides further granularity of access control by enabling the administrator to control the source and destination address, destination port, AND protocol. By implementing this level of access control an attacker would not be allowed access to other hosts residing on the protected network. Additionally, the TOE maintains "state" information of all approved connections. This function ensures that each packet arriving at a TOE interface purporting to be part of an approved connection through or to the TOE, is checked against a current and valid list of connection parameters (e.g. for a TCP/IP connection; source and destination address, ports, SYN and ACK numbers, flags, etc.) prior to allowing the packet through the TOE. The TOE requires successful authentication through a protected communication path (with account lock-out capability) to the TOE prior to gaining access to certain services on or mediated by the TOE. By implementing "protected" authentication to gain access to these services, an attacker's opportunity to successfully conduct a man-in-the-middle and/or password guessing attack is greatly reduced. Lastly, the TSF must ensure that all configured enforcement functions (authentication, access control rules, etc.) must be invoked prior to allowing a user to gain access to TOE or TOE mediated services. The TOE restricts the ability to modify the security attributes associated with access control rules and access to authenticated and unauthenticated services, etc to the Security Administrator. This feature ensures that no other user can modify the information flow policy to bypass the intended TOE security policy.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.UNIDENTIFIED_ACTIONS The administrator may fail to notice potential security violations, thus limiting the administrator's ability to identify and take action against a possible security breach.	O.AUDIT_REVIEW The TOE will provide the capability to selectively view audit information, and alert the administrator of identified potential security violations.	O.AUDIT_REVIEW (FAU_SAA.1-NIAP-0407, FAU_ARP.1, FAU_SAR.1, FAU_SAR.3) helps to mitigate this threat by providing the Security Administrator with a required minimum set of configurable audit events that could indicate a potential security violation. By configuring these auditable events, the TOE monitors the occurrences of these events (e.g. set number of authentication failures, set number of information policy flow failures, self-test failures, etc.) and immediately notifies all TOE administrators once an event has occurred or a set threshold has been met. If a potential security violation has been detected, the TOE displays a message that identifies the potential security violation to all administrative consoles. The consoles include the local TOE console and any active remote administrative sessions. If an administrator is not currently logged into the TOE, the message is stored and immediately displayed the next time an administrator logs into the TOE. This message is displayed to all administrative roles and will remain on the screen for each administrative role until each administrative role acknowledges the message. In addition to displaying the potential security violation, the message must contain all audit records that generated the potential security violation. By enforcing the message content and display, this objective provides assurance that a TOE administrator will be notified of a potential security violation. The TOE can also be configured to generate an audible alarm, which may alert administrators who are not sitting at their administrative workstation or console. The TOE also requires an Audit Administrative role. This role is restricted to Audit record review and the deletion of the audit trail for maintenance purposes. A search and sort capability provides an efficient mechanism for the Audit Administrator to view pertinent audit information.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.UNKNOWN_STATE When the TOE is initially started or restarted after a failure, design flaws, improper TOE configurations may cause the security state of the TOE to be unknown.	O.MAINT_MODE The TOE shall provide a mode from which recovery or initial startup procedures can be performed. O.CORRECT_ TSF_OPERATION The TOE will provide the capability to test the TSF to ensure the correct operation of the TSF in its operational environment. O.SOUND_DESIGN The design of the TOE will be the result of sound design principles and techniques; the design of the TOE, as well as the design principles and techniques, are adequately and accurately documented. O.ROBUST_ADMIN_GUIDANCE The TOE will provide administrators with the necessary information for secure delivery and management.	O.SOUND_DESIGN (ADV_SPM.1) works to mitigate this threat by requiring that the TOE developers provide accurate and complete design documentation of the security mechanisms in the TOE, including a security model. By providing this documentation, the possible security states of the TOE at startup or restart after failure should be documented and understood, thereby reducing the possibility that the TOE's security state could be unknown to users of the TOE. O.MAINT_MODE (FPT_RCV.1) helps to mitigate this threat by ensuring that the TOE does not continue to operate in an insecure state when a hardware or software failure occurs. After a failure, the TOE enters a state that disallows traffic flow and requires an administrator to follow documented procedures to return the TOE to a secure state. O.CORRECT_TSF_OPERATION (FPT_TST_EXP.4, FPT_TST_EXP.5), counters this threat by ensuring that the TSF runs a suite of tests to successfully demonstrates the correct operation of the TSF's underlying abstract machine (hardware and software), the TSF, and the TSF's cryptographic components at initial startup of the TOE. In addition to ensuring that the TOE's security state can be verified, the Security Administrator can verify the integrity of the TSF's data and stored code as well as the TSF's cryptographic data and stored code. O.ROBUST_ADMIN_GUIDANCE (ADO_IGS.1, AGD_ADM.1) provides administrative guidance for the secure start-up of the TOE as well as guidance to configure and administer the TOE securely. This guidance provides administrators with the information necessary to ensure that the TOE is started and initialized in a secure manor. The guidance also provides information about the corrective measure necessary when a failure occurs (i.e., how to bring the TOE back into a secure state).
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 system.	O.DISPLAY_BANNER The TOE will display an advisory warning regarding use of the TOE.	O.DISPLAY_BANNER (FTA_TAB.1) satisfies this policy by ensuring that the TOE displays a Security Administrator configurable banner that provides all users with a warning about the unauthorized use of the TOE.

Threat/Policy	Objectives Addressing the Threat	Rationale
P.ACCOUNTABILITY The authorized users of the TOE shall be held accountable for their actions within the TOE.	O.AUDIT_GENERATION The TOE will provide the capability to detect and create records of security-relevant events associated with users. O.TIME_STAMPS The TOE shall provide reliable time stamps and the capability for the administrator to set the time used for these time stamps. O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate	O.AUDIT_GENERATION (FAU_GEN.1-NIAP-0410, FAU_GEN.2-NIAP-0410, FIA_USB.1, FAU_SEL.1-NIAP-0407) addresses this policy by providing the Security Administrator with the capability of configuring the audit mechanism to record the actions of a specific user, or review the audit trail based on the identity of the user. Additionally, the administrator's ID is recorded when any security relevant change is made to the TOE (e.g. access rule modification, start-stop of the audit mechanism, establishment of a trusted channel, etc.). O.TIME_STAMPS (FPT_STM.1, FMT_MTD.1(3)) plays a role in supporting this policy by requiring the TOE to provide a reliable time stamp (configured locally by the Security Administrator or via an external NTP server). The audit mechanism is required to include the current date and time in each audit record. All audit records that include the user ID, will also include the date and time that the event occurred. O.ROBUST_TOE_ACCESS (FIA_UID.2, FIA_UAU_EXP.5) supports this policy by requiring the TOE to identify and authenticate all authorized users prior to allowing any TOE access or any TOE mediated access on behalf of those users. While the user ID of authorized users can be assured, since they are authenticated, this PP allows unauthenticated users to access the TOE and the identity is then a presumed network identifier (e.g., IP address).
P.ADMIN_ACCESS Administrators shall be able to administer the TOE both locally and remotely through protected communications channels.	O.ADMIN_ROLE The TOE will provide an administrator role to isolate administrative actions. O.TRUSTED_PATH The TOE will provide a means to ensure users are not communicating with some other entity pretending to be the TOE, and that the TOE is communicating with an authorized IT entity and not some other entity pretending to be an authorized IT entity.	O.ADMIN_ROLE (FMT_SMR.2) supports this policy by requiring the TOE to provide mechanisms (e.g., local authentication, remote authentication, means to configure and manage the TOE both remotely and locally) that allow remote and local administration of the TOE. This is not to say that everything that can be done by a local administrator must also be provided to the remote administrator. In fact, it may be desirable to have some functionality restricted to the local administrator (e.g., setting the ruleset). O.TRUSTED_PATH (FTP_TRP.1(1), FTP_TRP.1(2), FTP_ITC.1(1), FTP_ITC.1(2)) satisfies this policy by requiring that each remote administrative session (all administrative roles) is authenticated and conducted via a secure channel. Additionally, all authorized IT entities (e.g. authentication/certificate servers, NTP servers) must adhere to the same requirements as the remote administrator.

Threat/Policy	Objectives Addressing the Threat	Rationale
P.CRYPTOGRAPHIC_FUNCTIONS The TOE shall provide cryptographic functions for its own use, including encryption/decryption and digital signature operations.	O.CRYPTOGRAPHIC_FUNCTIONS The TOE shall provide cryptographic functions for its own use, including encryption/decryption and digital signature operations.	O.CRYPTOGRAPHIC_FUNCTIONS implements this policy, requiring a combination of FIPS-validation and non-FIPS-validated cryptographic mechanisms that are used to provide encryption/decryption services, as well as digital signature functions. Functions include symmetric encryption and decryption, digital signatures, as well as key generation and establishment functions.
P.CRYPTOGRAPHY_VALIDATED Where the TOE requires FIPS-approved security functions, only NIST FIPS validated cryptography (methods and implementations) are acceptable for key management (i.e.; generation, access, distribution, destruction, handling, and storage of keys) and cryptographic services (i.e.; encryption, decryption, signature, hashing, key distribution, and random number generation services).	O.CRYPTOGRAPHY_VALIDATED The TOE shall use NIST FIPS 140-2 validated cryptomodules for cryptographic services implementing FIPS-approved security functions and random number generation services used by cryptographic functions. O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated or upon completion of a function that residual biometric data could not be reused.	O.CRYPTOGRAPHY_VALIDATED satisfies this policy by requiring the TOE to implement NIST FIPS validated cryptographic services. These services will provide confidentiality and integrity protection of TSF data while in transit to remote parts of the TOE. O.RESIDUAL_INFORMATION satisfies this policy by ensuring that cryptographic data are cleared from resources that are shared between users. Keys must be zeroized according to FIPS 140-2.
P.VULNERABILITY_ ANALYSIS_TEST The TOE must undergo appropriate independent vulnerability analysis and penetration testing to demonstrate that the TOE is resistant to an attacker possessing a medium attack potential.	O.VULNERABILITY_ANALYSIS_ TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	O.VULNERABILITY_ANALYSIS_TEST (AVA_VLA.3) 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.

Table 5 - Security Objectives to Threats and Policies Mappings

6.2 Rationale for the Security Objectives and Security Functional Requirements for the Environment

All but one of the security objectives for the environment, OE.CRYPTANALYTIC, are restatements of an assumption found in Section 3. Therefore, those security objectives for the non-IT environment trace to the assumptions trivially and are suitable for covering the assumptions.

The IT environment security objective OE.CRYPTANALYTIC is necessary to play a role in countering the threat T.CRYPTO_COMPROMISE. This IT environment security objective ensures that the cryptographic methods used in the IT environment are interoperable with the mechanisms provided by the TOE. The IT environment's cryptographic methods should be independently validated to be FIPS 140-2 compliant. OE.CRYPTANALYTIC maps to the IT environmental iterated requirements FPT_ITC.1 (ensuring that encryption is used on the communication channel between authorized IT entities and the TOE), and FPT_TRP (ensuring that an administrator can be assured that they are communicating with the TOE).

6.3 Rationale for TOE Security Requirements

Objective	Requirements Addressing the Objective	Rationale
O.ROBUST_ADMIN_GUIDANCE The TOE will provide administrators with the necessary information for secure delivery and management.	ADO_DEL.2 AGD_ADM.1 AVA_MSU.2 ADO_IGS.1 AGD_USR.1	ADO_DEL.2 ensures that the administrator is provided documentation that instructs them how to ensure the delivery of the TOE, in whole or in parts, has not been tampered with or corrupted during delivery. This requirement ensures the administrator has the ability to begin their TOE installation with a *clean* (e.g., malicious code has not been inserted once it has left the developer's control) version of the TOE, which is necessary for secure management of the TOE. The ADO_IGS.1 requirement ensures the administrator has the information necessary to install the TOE in the evaluated configuration. Often times a vendor's product contains software that is not part of the TOE and has not been evaluated. The Installation, Generation and Startup (IGS) documentation ensures that once the administrator has followed the installation and configuration guidance the result is a TOE in a secure configuration. The AGD_ADM.1 requirement mandates the developer provide the administrator with guidance on how to operate the TOE in a secure manner. This includes describing the interfaces the administrator uses in managing the TOE, security parameters that are configurable by the administrator, how to configure the TOE's ruleset and the implications of any dependencies of individual rules. The documentation also provides a description of how to setup and review the auditing features of the TOE. The AGD_USR.1 is intended for non-administrative users, but could be used to provide guidance on security that is common to

Objective	Requirements Addressing the Objective	Rationale
		both administrators and non-administrators (e.g., password management guidelines). The use of the authentication mechanism would not have to be repeated in the administrator's guide. AVA_MSU.2 ensures that the guidance documentation is complete and can be followed unambiguously to ensure the TOE is not mis-configured in an unsecure state due to confusing guidance.

O.ADMIN_ROLE The TOE will provide administrator roles to isolate administrative actions.	FMT_SMR.2	FMT_SMR.2 requires that three roles exist for administrative actions: the Security Administrator, who is responsible for configuring the TOE's security policies; the Cryptographic Administrator, who is restricted to managing the security data that is critical to the cryptographic operations; and the Audit Administrator, who is restricted to reading and deleting the audit trail. The TSF is able to associate a human user with one or more roles and these roles isolate administrative functions in that the functions of these roles do not overlap. The functionality of the roles, as defined by this PP, is predicated on the notion that once the TOE has been setup and is running in a stable configuration the Security Administrator would not be required to frequently administer the TOE. The Audit Administrator will probably be logging into the TOE most often to review the audit trail. Restricting the Audit Administrator's capabilities thus reduces the

Objective	Requirements Addressing the Objective	Rationale
		potential harm that could occur due to an error, or the execution of malicious code.
O.AUDIT_GENERATION The TOE will provide the capability to detect and create records of security-relevant events associated with users.	FAU_GEN.1-NIAP-0410 FAU_GEN.2-NIAP-0410 FIA_USB.1 FAU_STG.3 FAU_SEL.1-NIAP-0407 FAU_STG.NIAP-0414-1-NIAP-0429	FAU_GEN.1-NIAP-0410 defines the set of events that the TOE must be capable of recording. This requirement ensures that the Security 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-NIAP-0410 ensures that the audit records associate a user identity with the auditable event. In the case of authorized users, the association is accomplished with the userid. In all other cases, the association is based on the source network identifier, which is presumed to be the correct identity, but cannot be confirmed since these subjects are not authenticated. FAU_SEL.1-NIAP-0407 allows the Security 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 resources consumed by the audit mechanism. FAU_STG.3 requires that the administrators are alerted when the audit trail exceeds a capacity threshold established by the Security Administrator. This ensures that the Security Administrator has the opportunity to manage the audit trail before it becomes full and the avoiding the possible loss of audit data. FAU_STG.NIAP-0414-1-NIAP-0429 allows the Security Administrator to configure the TOE so that if the audit trail does become full, either the TOE will prevent any events from occurring (other than actions taken by the Security Administrator or Audit Administrator) that would generate an audit record (e.g., depending on the FAU_SEL.1-NIAP-0407 configuration, traffic may no longer flow through the TOE) or the audit mechanism will overwrite the oldest audit records with new records. 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 authorized users, since the identity of unauthenticated users cannot be confirmed. Therefore, the audit trail may not always have the proper identity of the subject that causes an audit record to be generated (e.g., presumed network address of an unauthenticated user may be a spoofed address).
O.AUDIT_PROTECTION The TOE will provide the capability to protect audit information.	FAU_STG.1-NIAP-0423 FAU_SAR.2 FAU_STG.NIAP-0414-1-NIAP-0429	FAU_SAR.2 restricts the ability to read the audit trail to the administrators, thus preventing the disclosure of the audit data to any other user. However, the TOE is not expected to prevent the disclosure of audit data if it has been archived or saved in another form (e.g., moved or copied to an ordinary file). The FAU_STG family dictates how the audit trail is protected. FAU_STG.1-NIAP-0423 restricts the ability to delete audit

Objective	Requirements Addressing the Objective	Rationale
	FAU_STG.3 FMT_MOF.1(2)	records to the Audit Administrator or if the option of overwriting old audit records is chosen by the Security Administrator in FAU_STG.NIAP-0414-1-NIAP-0429, the audit data may be deleted/overwritten. This helps ensure that audit records are kept until the Audit Administrator deems they are no longer necessary. This requirement also ensures that no one has the ability to modify audit records (e.g., edit any of the information contained in an audit record). This ensures the integrity of the audit trail is maintained. FMT_MOF.1(2) restricts the capability to modify the behavior of the audit and alarm functions to the Security Administrator. While the Audit Administrator has the capability to choose how they will review the audit trail, they do not have the capability to select what events are audited. This requirement ensures that only the Security Administrator can turn audit on or off, thus ensuring user's actions are audited according to a site-defined policy.
O.AUDIT_REVIEW The TOE will provide the capability to selectively view audit information, and alert the administrator of identified potential security violations.	FAU_SAA.1-NIAP-0407 FAU_ARP.1 FAU_ARP_ACK_EXP.1 FMT_MOF.1(3) FAU_SAR.3 FAU_SAR.1 FMT_MOF.1(4) FMT_MOF.1(5)	FAU_SAA.1-NIAP-0407 defines the events that indicate a potential security violation and will generate an alarm. The triggers for these events are configurable, for the most part, by the Security Administrator. The exception is that any failure of the TSF self-tests will generate an alarm. FAU_ARP.1 requires that the alarm be displayed at the local administrative console and at the remote administrative console(s) when an administrative session exists. For the latter, the alarm is sent to each role either during an established session or upon session establishment. This is required to ensure that no matter which role an administrator logs into the alarm will be received as soon as possible. This requirement also dictates the information that must be displayed with the alarm. The potential security violation is identified in the alarm, as are the contents of the audit records of the events that accumulated and triggered the alarm. The information in the audit records is necessary it allows the administrators to react to the potential security violation without having to search through the audit trail looking for the related events. The TOE can also be configured to generate an audible alarm, which notifies administrators that are not attending their workstations of the potential violation. FAU_ARP_ACK_EXP.1 requires that the alarm be displayed at the local administrative console until it is acknowledged by an administrator, and at the remote administrative console(s) until it has been acknowledged by an administrator acting in each of the administrative roles. This ensures that the alarm message will not be obstructed and the administrators will be alerted of a potential security violation. The audible alarm, if configured, sounds continuously until acknowledged by an administrator. FAU_SAR.1 provides the administrators with the capability to read all the audit data contained in the audit trail. This requirement also mandates the audit information be presented in a manner that is suitable for the administrators to interpret the audit trail, which is subject to interpretation. It is expected that the audit information be presented in such a way that the administrators can examine an audit record and have the appropriate information (that required by FAU_GEN.2-NIAP-0410) presented together to facilitate the analysis of the audit review. FAU_SAR.3 complements FAU_SAR.1 by providing the administrators the flexibility to specify criteria that can be used to

Objective	Requirements Addressing the Objective	Rationale
		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 userid and source subject identity, so that the actions of a user can be readily identified and analyzed. The criteria also includes a destination subject identity so the administrators can determine what network traffic is destined for an individual machine. Allowing the administrators to perform searches or sort the audit records based on dates, times, subject identities, destination service identifier, or transport layer protocol provides the capability to extract the network activity to what is pertinent at that time in order facilitate the administrator's review. Being able to search on the destination service identifier affords the administrators the opportunity to see what traffic is destined for a service (e.g., TCP port) or set of services regardless of where the traffic originated. 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. For example, if the administrators wanted to see what network traffic was destined for the set of TCP ports 1-1024, they would be able to have the audit data presented in such a way that all the traffic for TCP port 1 was grouped together, all the traffic for port 2 was grouped together and so on. The criteria includes the rule identity that determines whether a packet was allowed or denied to flow. This provides the administrators to determine what network traffic a given rule is governing. FMT_MOF.1(3) restricts the ability to control the behavior of the audit and alarm mechanism to the administrators. The Security Administrator is the only user that controls the behavior of the events that generate alarms. FMT_MOF.1(4) provides the administrators "read only" access to the audit records and prohibits access to all other users. Additionally the administrators are provided the capability to "search and sort" audit on defined criteria. This capability expedites problem resolution analysis. FMT_MOF.1(5) ensures that only an administrators can "enable or disable" the security alarms. This requirement works with FMT_MOF.1(4) to provide detailed granularity to the administrator when determining which actions constitute a security violation
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.	ACM_CAP.4 ACM_SCP.2 ALC_DVS.1 ALC_FLR.2 ALC_LCD.1 ACM_AUT.1	ACM_CAP.4 contributes to this objective by requiring the developer 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_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.

Objective	Requirements Addressing the Objective	Rationale
		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. 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.
O.CORRECT_ TSF_OPERATION The TOE will provide the capability to test the TSF to ensure the correct operation of the TSF in its operational environment.	FPT_TST_EXP.4, FPT_TST_EXP.5	O_CORRECT_TSF_OPERATION requires two security functional requirements in the FPT class, FPT_TST. These functional requirements provide the end user with the capability to ensure the TOE's security mechanisms continue to operate correctly in the field. FPT_TST_EXP.4 has been created to ensure end user tests exist to demonstrate the correct operation of the security mechanisms required by the TOE that are provided by the hardware and that the TOE's software and TSF data has not been corrupted. Hardware failures could render a TOE's software ineffective in enforcing its security policies and this requirement provides the end user the ability to discover any failures in the hardware security mechanisms. FPT_TST_EXP.4 is necessary to ensure 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.
O.CRYPTOGRAPHY_VALIDATED The TOE shall use NIST FIPS 140-2 validated cryptomodules for cryptographic services implementing FIPS-approved security functions and random number generation services used by cryptographic functions.	FCS_BCM_EXP.1 FCS_CKM.1 FCS_COP_EXP.5 FCS_COP_EXP.6	This objective deals with the issue of using FIPS 140-2-approved cryptomodules in the TOE. A cryptomodule, as used in the components, is a module that is FIPS 140-2 validated (in accordance with FCS_BCM_EXP.1); the cryptographic functionality implemented in that module are FIPS-approved security functions that have been validated; and the cryptographic functionality is available in a FIPS-approved mode of the cryptomodule. This objective is distinguished from O.CRYPTOGRAPHIC_FUNCTIONS in that this deals only with a requirement to use FIPS 140-2-validated cryptomodules where the TOE requires such functionality; it does not dictate the specific functionality that is to be used. FCS_BCM_EXP.1 is an explicit requirement that specifies not

Objective	Requirements Addressing the Objective	Rationale
		only that cryptographic functions that are FIPS-approved must be validated by FIPS, but also what NIST FIPS rating level the cryptographic module must satisfy. The level specifies the degree of testing of the module. The higher the level, the more extensive the module is tested. FCS_CKM.1 mandates that the cryptomodule must generate key, and that this key generation must be part of the FIPS-validated cryptomodule. FCS_COP_EXP.5 and FCS_COP_EXP.6 are similar in that they require that any random number generation and hashing functions, respectively, are part of a FIPS-validated cryptographic module. These requirements do not mandate that the functionality is generally available, but only that it be implemented in a FIPS-validated module should other cryptographic functions need these services.
O.CRYPTOGRAPHIC_FUNCTIONS The TOE shall provide cryptographic functions for its own use, including encryption/decryption and digital signature operations.	FCS_CKM.1 FCS_CKM_SYM_EXP.1 FCS_CKM_ASYM_EXP.1 FCS_CKM.4 FCS_COP_EXP.2 FCS_COP_EXP.3	The FCS requirements used in this PP satisfy this objective by levying requirements that ensure the cryptographic standards include the NIST FIPS publications (where possible) and NIST approved ANSI standards. The intent is to have the satisfaction of the cryptographic standards be validated through a NIST FIPS 140 validation. In contrast to O.CRYPTOGRAPHY_VALIDATED, this objective is to provide cryptographic functionality that is used by the TOE. The core functionality to be supported is encryption/decryption using a symmetric algorithm, and digital signature generation and verification using asymmetric algorithms. Since these operations involve cryptographic keys, how the keys are generated and/or otherwise obtained have to also be specified. FCS_CKM.1 is a requirement that a cryptomodule generate symmetric keys. Such keys are used by the AES encryption/decryption functionality specified in FCS_COP_EXP.2. Another way of obtaining key material for symmetric algorithms is through cryptographic key establishment, as specified in FCS_CKM_SYM_EXP.1 and for asymmetric algorithms specified in FCS_CKM_ASYM_EXP.1. Key establishment has two aspects: key agreement and key distribution. Key agreement occurs when two entities exchange public data yet arrive at a mutually shared key without ever passing that key between the two entities (for example, the Diffie-Hellman algorithm). Key distribution occurs when the key is transmitted from one entity to the TOE. If the entity is electronic and a protocol is used to distribute the key, it is referred to in this PP as “Key Transport”. If the key is loaded into the TOE it can be loaded electronically (e.g., from a floppy drive, smart card, or electronic keyfill device) or manually (e.g., typed in). One or more of these methods must be selected. FCS_CKM.4 provides the functionality for ensuring key and key material is zeroized. This applies not only to key that resides in the TOE, but also to intermediate areas (physical memory, page files, memory dumps, etc.) where key may appear. As previously mentioned FCS_COP_EXP.2 specifies that AES be used to perform encryption and decryption operations. FCS_COP_EXP.3 gives two options for providing the digital

Objective	Requirements Addressing the Objective	Rationale
		signature capability; these requirements also contain requirements for obtaining and generating the domain parameters and key for each of the algorithms.
O.DISPLAY_BANNER The TOE will display an advisory warning regarding use of the TOE.	FTA_TAB.1	FTA_TAB.1 meets this objective by requiring the TOE display a Security Administrator defined banner before a user can establish an authenticated session. This banner is under complete control of the Security Administrator in which they specify any warnings regarding unauthorized use of the TOE and remove any product or version information if they desire.
O.DOCUMENT_KEY_ LEAKAGE The bandwidth of channels that can be used to compromise key material shall be documented.	AVA_CCA_EXP.2	AVA_CCA_EXP.2 requires that a covert channel analysis be performed on the entire TOE to determine the bandwidth of possible cryptographic key leakage. While there are no requirements to limit the bandwidth, the results of this analysis will provide useful guidance on what the specified lifetime of the cryptographic keys should be in order to reduce the damage due to a key compromise.
O.THOROUGH_FUNCTIONAL_ TESTING The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.	ATE_COV.2 ATE_FUN.1 ATE_DPT.2 ATE_IND.2	In order to satisfy O.THOROUGH_FUNCTIONAL_ TESTING, the ATE class of requirements is necessary. 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.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.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. 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.
O.MAINT_MODE The TOE shall provide a mode from which recovery or initial startup procedures can be performed.	FPT_RCV.1	This objective is met by using the FPT_RCV.1 requirement, which 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

Objective	Requirements Addressing the Objective	Rationale
		required in some cases, does not meet the intent of this requirement.
O.MANAGE The TOE will provide all the functions and facilities necessary to support the administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.	FMT_MSA.1 FMT_MSA.3-NIAP-0409(1) FMT_MSA.3-NIAP-0409 (2) FMT_MOF.1(1) FMT_MOF.1(2) FMT_MOF.1(3) FMT_MOF.1(4) FMT_MOF.1(5) FMT_MOF.1(6) FMT_MOF.1(76) FMT_MTD.1(1) FMT_MTD.1(2) FMT_MTD.1(3) FMT_MTD.1(4) FAU_SAR.1 FAU_SAR.2 FAU_SAR.3 FAU_SEL.1-NIAP-0407 FAU_STG.1-NIAP-0423 FAU_STG.3 FAU_STG.NIAP-0414-1-NIAP-0429 FAU_ARP_ACK_EXP.1	The FMT requirements are used to satisfy this management objective, as well as other objectives that specify the control of functionality. The requirement's rationale for this objective focuses on the administrator's capability to perform management functions in order to control the behavior of security functions. FMT_MSA.1-NIAP-0409 provides the Security Administrator the capability to manipulate the security attributes to facilitate the construction of the ruleset. An example of this would be to group a set of service identifiers that are to have the same rule applied, rather than having to specify a separate rule for each service identifier. FMT_MSA.3-NIAP-0409 requires that by default, the TOE does not allow an information flow, rather than allowing information flows until a rule in the ruleset disallows it. FMT_MOF.1(2) and FMT_MSA.3-NIAP-0409(2) are related to the services provided by FAU_UAU.1(1) and provide the Security Administrator control as to the availability of these services. FMT_MOF.1(2) provides the ability to enable or disable the TOE services to the Security Administrator. FMT_MSA.3-NIAP-0409(2) requires that these services by default are disabled. Since the Security Administrator must explicitly enable these services it ensures the Security Administrator is aware that they are running. This requirement does afford the Security Administrator the capability to override this restrictive default and allow the services to be started whenever the TOE reboots or is restarted. FMT_MOF.1(1) is used to ensure the administrators have the ability to invoke the TOE self-tests at any time. The ability to invoke the self-tests is provided to all administrators. The Security Administrator is able to modify the behavior of the tests (e.g., select when they run, select a subset of the tests). FMT_MOF.1(3) specifies the ability of the administrators to control the security functions associated with audit and alarm generation. The ability to control these functions has been assigned to the appropriate administrative roles. FMT_MOF.1(7) This requirement limits the ability to manipulate the values that are used in the FRU_RSA.1(2) requirements to the Security Administrator. The Security Administrator is provided the capability to assign the network identifier(s) they wish to place resource restrictions on and allows them to also specify over what period of time those quota limitations are in place. FMT_MOF.1(4) provides the administrators "read only" access to the audit records and prohibits access to all other users. Additionally the administrators are provided the capability to "search and sort" audit on defined criteria. This capability expedites problem resolution analysis. FMT_MOF.1(5) ensures that only an administrators can "enable or disable" the security alarms. This requirement works with FMT_MOF.1(4) to provide detailed granularity to the administrator when determining which actions constitute a security violation

Objective	Requirements Addressing the Objective	Rationale
		FMT_MOF.1(6) limits the ability to enable or disable unauthenticated TOE services for both IP based networks and non-IP based networks to the Security Administrator. These TOE services would be available to appropriate network users at the discretion of the Security Administrator. FMT_MOF.1(7) provides the Security Administration configuration control of the allocation of connection-oriented TOE resources. This requirement provides the Security Administrator with a capability to thwart possible external "resource allocation" attacks on the TOE. The requirement FMT_MTD.1(1) is intended to be used by the ST author, with possible iterations, to address TSF data that has not already been specified by other requirements. This is necessary because the ST author may add TSF data in assignments that cannot be addressed apriori by the PP authors. FMT_MTD.1(2) provides the Cryptographic Administrator, and only the Cryptographic Administrator, the ability to modify the cryptographic security data. This allows the Cryptographic Administrator to change the critical data that affects the TOE's ability to perform its cryptographic functions properly. FMT_MTD.1(3) provides the capability of setting the date and time that is used to generate time stamps to the Security Administrator or an authorized IT entity. It is important to allow this functionality, due to clock drift and other circumstances, but the capability must be restricted. An authorized IT entity is allowed in the selection made by the ST author to take in account the use of an NTP server or some other service that provides time information without human intervention. FMT_MTD.1(4) provides the Security Administrator the capability to manage the TOE's ruleset. This capability is restricted to only the Security Administrator and allows them to create, view, modify and delete the rules that comprise the ruleset. FAU_SAR.1 ensures that the Audit Administrator has the capability to review the audit records and that they are presented in a manner that is suitable for review (e.g., the Audit Administrator can construct a sequence of events provided the necessary events were audited). FAU_SAR.2 restricts the ability to read the audit records to the administrators. This capability exists for the Security and Crypto administrators to help facilitate any trouble shooting that they may have to perform. FAU_SAR.3 provides the administrators with the ability to selectively review the contents of the audit trail based on established criteria. This capability allows the administrators to focus their audit review to what is pertinent at that time. FAU_STG.1-NIAP-0423 specifies that only the Audit Administrator can delete the audit trail. This prevents the accidental or intentional deletion of the audit trail by administrators acting in another role. FAU_STG.3 provides the Security Administrator the capability to establish a threshold of audit trail capacity, that when reached an

Objective	Requirements Addressing the Objective	Rationale
		alarm will be generated. If the audit trail becomes full FAU_STG.NIAP-0414-1-NIAP-0429 provides the Security Administrator the option of having the TOE prevent auditable events from occurring, or having the TOE overwrite the oldest audit records. While the option of overwriting old audit records does not technically prevent audit data loss, it is provided to the Security Administrator as an option to prevent a possible denial-of-service. FAU_ARP_ACK_EXP.1 contributes to this objective in that it requires the administrators to acknowledge an alarm before it is no longer displayed. Without this requirement an alarm display message may be overwritten or lost without an administrator being aware of the alarm condition.
		FAU_SEL.1-NIAP-0407 provides the Security Administrator the ability to define what events will be included or excluded from the list of audited events. This allows a site to audit only those events that are of interest to them and reduces the amount of unwanted audit data that is collected.
O.MEDIATE The TOE must mediate the flow of information between sets of TOE network interfaces or between a network interface and the TOE itself in accordance with its security policy.	FDP_IFF.1-NIAP-0417(1) FDP_IFF.1-NIAP-0417(2) FDP_IFC.1(1) FDP_IFC.1(2) FMT_REV.1 FPT_RVM.1	The FDP_IFF and FDP_IFC requirements were chosen to define the policies, the subjects, objects, and operations for how and when mediation takes place. FDP_IFC.1(1) and FDP_IFC.1(2) define the subjects, information (e.g., objects) and the operations that are performed with respect to the three information flow policies. FDP_IFC.1(1), the subjects are defined to be a source subject, which is the TOE's network interface on which a packet is received, and a destination subject, which is the TOE's network interface on which the packet is destined. The information flow control requirements are not well suited for a firewall. This subject determination was made since the TOE network interfaces are something the TOE has control over (e.g., the administrator has the ability to assign network identifiers to these interfaces, which is a critical component in the mediation decision) and rules could be identified in FDP_IFF.1-NIAP-0417(1) that make sense with respect to mediation of information. The alternative was to classify the sender and receiver of the data packets as subjects, but the sender and receiver are not under the control of the TOE and would not make sense to perform mediation under those circumstances. The objects in this policy are defined to be the network packets, since that is the entity that the operations are performed on. Those operations are to pass the information if the mediation allows the flow, otherwise the packet is dropped. FDP_IFF.1-NIAP-0417(1) is used to specify the policy of unauthenticated traffic flowing through the TOE. This requirement ensures that the network traffic is mediated (i.e., the ruleset is used) even though the subjects have not been authenticated. This requirement also mandates the TOE perform stateful inspection of the packets to determine if they should be allowed to flow through the TOE. The stateful inspection attributes are not intended to specifiable by the Security Administrator, rather these attributes are to be "managed" and mediated internally by the TOE. FDP_IFC.1(2) defines subjects for the unauthenticated access to any services the TOE provides. This is different from the other policies in that the TOE mediates access to itself, rather than determining if information should be allowed to flow through the

Objective	Requirements Addressing the Objective	Rationale
		TOE. The destination subject is defined to be the TOE, and the source subject is the TOE interface on which a network packet is received. The information remains the same, a network packet, and the operations are limited to accept or reject the packet. FDP_IFF.1-NIAP-0417(2) provides the rules that apply to the unauthenticated use of any services provided by the TOE. ICMP is the only service that is required to be provided by the TOE, and the security attributes associated with this protocol allow the Security Administrator to specify what degree the ICMP traffic is mediated (i.e., the ICMP message type and code). The ST author could specify other services they wish their TOE implementation to provide, and if they do so, they should also specify the security attributes associated with the additional services. FMT_REV.1 is a management requirement that affords the Security Administrator the ability to immediately revoke user's ability to send network traffic through the TOE. If the Security Administrator revokes a user's access (e.g., via a rule in the ruleset, revoking an administrative role from a user) the TOE will immediately enforce the new Security Administrator defined "policy". FPT_RVM.1 ensures that packets that flow through the TOE, or those that are destined for the TOE are mediated with respect to the identified policies. Each TSF interface that operates on subjects or objects that are identified in the explicit policies, or operates on TSF data or security attributes, must ensure that the operation is checked against the explicit and implicit security policies defined in this PP. If any TSF interface allows unchecked access to any of these resources, then the TOE cannot be relied upon to enforce the security policies.
O.REPLAY_DETECTION The TOE will provide a means to detect and reject the replay of TSF data and security attributes.	FPT_RPL.1	The O.REPLAY_DETECTION objective is satisfied by the requirement FPT_RPL.1, which requires the TOE to not only detect, but to also reject the attempted replay of TSF data, and security attributes. This requirement also requires the TOE to audit the detection of replay, 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 as desired by the administrators.
O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.	FDP_RIP.2 FCS_CKM.4	FDP_RIP.2 is used to ensure the contents of resources are not available to subjects other than those explicitly granted access to the data. For this TOE it is critical that the memory used to build network packets is either cleared or that some buffer management scheme be employed to prevent the contents of a packet being disclosed in a subsequent packet (e.g., if padding is used in the construction of a packet, it must not contain another user's data or TSF data). 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.
O.RESOURCE_SHARING The TOE shall provide mechanisms that mitigate attempts to exhaust connection-oriented resources provided by the TOE (e.g., entries in a connection state table; TCP	FRU_RSA.1(1) FRU_RSA.1(2) FMT_MTD.2(1)	While an availability security policy does not explicitly exist, FRU_RSA.1 was used to mitigate potential resource exhaustion attempts. FRU_RSA.1(1) was used to reduce the impact of an attempt being made to exhaust the transport-layer representation (e.g., attempt to make the TSF unable to respond to connection-oriented requests, such as SYN attacks). This requirement allows

Objective	Requirements Addressing the Objective	Rationale
connections used by proxies).	FMT_MTD.2(2) FMT_MOF.1(7)	the administrator to specify the time period in which when maximum quota (which is defined by the ST) is met or surpassed, an ST defined action is to take place, which is specified in FMT_MTD.2(1). These two requirements together help limit the resources that can be utilized by the general population of users as a whole. An issue with treating all the users the same is that legitimate users may not be able to establish connections due to the connection table entries being exhausted. Therefore FRU_RSA.1(2) is also included. FRU_RSA.1(2) is more specific in that attempts to exhaust the connection-oriented resources by a single network address, or a set of network addresses can be controlled. This affords the administrator a finer granularity of control than FRU_RSA.1(1). FRU_RSA.1(2) has the advantage of providing the Security Administrator with the ability to define the maximum number of resources a particular address or set of addresses can use over a specified time period. This requirement works in conjunction with FMT_MTD.2(2) which restricts the ability to set the quotas to the security administrator and allows for the ST author to assign what actions will take place once the quotas are met or surpassed. This iteration of FPT_RSA.1 makes it less likely that a legitimate user of the TOE will be denied access due to resource exhaustion attempts. FMT_MOF.1(7) restricts the ability to assign the single network address or set of network addresses used in FRU_RSA.1(2) to the Security Administrator. This is in keeping with the TOE's notion of the Security Administrator is responsible for configuring the TOE's policy enforcement mechanisms.
O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure.	FPT_SEP.2 FPT_RVM.1 FTP_ITC.1(1), FTP_ITC.1(2) FTP_TRP.1(1), FTP_TRP.1(2)	FPT_SEP was chosen to ensure the TSF provides a domain that protects itself from untrusted users. If the TSF cannot protect itself it cannot be relied upon to enforce its security policies. FPT_SEP.1 could have been used to address the previous notion, however, FPT_SEP.2 was used to require that the *cryptographic module* be provided its own address space. This is necessary to reduce the impact of programming errors in the remaining portions of the TSF on the cryptographic module. The inclusion of FPT_RVM.1 ensures that the TSF makes policy decisions on all interfaces that perform operations on subjects and objects that are scoped by the policies. Without this non-bypassability requirement, the TSF could not be relied upon to completely enforce the security policies, since an interface(s) may otherwise exist that would provide a user with access to TOE resources (including TSF data and executable code) regardless of the defined policies. This includes controlling the accessibility to interfaces, as well as what access control is provided within the interfaces. FTP_ITC.1(1), FTP_ITC.1(2) and FTP_TRP.1(1), FTP_TRP.1(2) are necessary for communication between the TOE and other trusted IT entities (e.g., authentication server, authorized IT entities) and the TOE and remote administrators. In order to protect TSF data and security attributes there is need for a trusted channel/trusted path. The trusted channel ensures that the authentication data that is supplied to the TOE is not compromised. It may be the case that the TOE relies upon an authorized IT entity to supply/manage TSF data (e.g., time stamp). If this is the case, the trusted channel ensures the TSF data is not compromised. The aspect of the trusted path that applies to this objective is FTP_TRP.1.3, which requires that the entire remote

Objective	Requirements Addressing the Objective	Rationale
		administrative session be protected. The protection of the communication path when TSF data is being transmitted is critical to the TSF maintaining a domain of execution that cannot be tampered or interfered with, thus resulting is a possible unauthorized disclosure or security policy failure.
O.SOUND_DESIGN The design of the TOE will be the result of sound design principles and techniques; the design of the TOE, as well as the design principles and techniques, are adequately and accurately documented.	ADV_FSP_EXP.1 ADV_HLD_EXP.1 ADV_INT_EXP.1 ADV_LLD_EXP.1 ADV_RCR.1 ADV_ARC_EXP.1 ADV_SPM.1	There are two different perspectives for this objective. One is from the developer's point of view and the other is from the evaluator's. 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_INT_EXP.1ensures 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's understanding of the interactions within the TSF, which in turn, potentially 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. The appropriate level of understanding is dictated by other assurance requirements in this PP (e.g., ATE_DPT.2, AVA_CCA_EXP.2, AVA_VLA.3). 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_FSP_EXP.1 requires that the interfaces to the TSF be completely specified. In this TOE, a complete specification of the network interface (including the network interface card) is critical in understanding what functionality is presented to untrusted users and how that functionality fits into the enforcement of security policies. Some network protocols have inherent flaws and users have the ability to provide the TOE with network packets crafted to take advantage of these flaws. The routines/functions that process the fields in the network protocols allowed (e.g., TCP, UPD, ICMP, any application level) must fully specified: the acceptable parameters, the errors that can be generated, and what, if any, exceptions exist in the processing. The functional specification of the hardware interface (e.g., network interface card) is also extremely critical. Any processing that is externally visible performed by NIC must be specified in the functional specification. Having a complete understanding of what is available at the TSF interface allows one to analyze this functionality in the context of design flaws. ADV_HLD_EXP.1 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 an SFR) 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.

Objective	Requirements Addressing the Objective	Rationale
		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 SFRs. 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. ADV_ARC_EXP.1 addresses the non-bypassability (FPT_RVM) and domain separation (FPT_SEP) aspects of the TSF, since these need to be analyzed differently from other functional requirements. 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 SFRs. 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.
O.SOUND_IMPLEMENTATION The implementation of the TOE will be an accurate instantiation of its design, and is adequately and accurately documented.	ADV_IMP.2 ADV_LLD_EXP.1 ADV_RCR.1 ADV_INT_EXP.1 ALC_TAT.1	While ADV_LLD_EXP.1 is used to aide in ensuring that the TOE's design is sound, it also contributes to ensuring the implementation is correctly realized from the design. It is expected that evaluators will use the low-level design as an aide in understanding the implementation representation. The low-level design requirements ensure the evaluators have enough information to intelligently analyze (e.g., the documented interface descriptions of the modules match the entry points in the module, error codes returned by the functions in the module are consistent with those identified in the documentation) the implementation and ensure it is consistent with the design. While evaluators have the ability to "negotiate" the subset in ADV_IMP.1, ADV_IMP.2 was chosen to ensure evaluators have full access to the source code. If the evaluators are limited in their ability to analyze source code they may not be able to determine the accuracy of the implementation or the adequacy of the documentation. Often times it is difficult for an evaluator to identify the complete sample of code they wish to analyze. Often times looking at code in one subsystem may lead the evaluator to discover code they should look at in another subsystem. Rather than require the evaluator to "re-negotiate" another sample of code, the complete implementation representation is required. When performing the activities associated with the ADV_INT_EXP.1requirement, the evaluators will ensure that the

Objective	Requirements Addressing the Objective	Rationale
		architecture of the implementation is modular and consistent with the architecture presented in the low-level design. Having a modular implementation provides the evaluators with the ability to more easily assess the accuracy of the implementation, with respect to the design. If the implementation is overly complex (e.g., circular dependencies, not well understood coupling, reliance on side-effects) the evaluator may not have the ability to assess the accuracy of the implementation. ALC_TAT.1 provides evaluators with information necessary to understand the implementation representation and what the resulting implementation will consist of. Critical areas (e.g., the use of libraries, what definitions are used, compiler options) are documented so the evaluator can determine how the implementation representation is to be analyzed. ADV_RCR.1 is used here to provide the correspondence of the lowest level of decomposition (e.g., source code) to the adjoining level, low-level design. The correspondence analysis is used by the evaluator as a tool when determining if the low-level design is correctly reflected in the implementation representation.
O.TIME_STAMPS The TOE shall provide reliable time stamps and the capability for the administrator to set the time used for these time stamps.	FPT_STM.1 FMT_MTD.1(3)	FPT_STM.1 requires that the TOE be able to provide reliable time stamps for its own use and therefore, partially satisfies this objective. Time stamps include date and time and are reliable in that they are always available to the TOE, and the clock must be monotonically increasing. FMT_MTD.1(3) satisfies the rest of this objective by providing the capability to set the time used for generating time stamps to either the Security Administrator, authorized IT entity, or both, depending on the selection made by the ST author. The authorized IT entity was included as an option for the possible use of an NTP server to set the TOE's time.
O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate	FTA_TSE.1 FIA_UID.2 FTA_SSL.1 FTA_SSL.2 FTA_SSL.3 AVA_SOF.1 FIA_AFL.1-NIAP-0425 FIA_ATD.1 FIA_UAU.1 FIA_UAU_EXP.5	FIA_UID.2 plays a small role in satisfying this objective by ensuring that every user is identified before the TOE performs any mediated functions. In some cases, the identification cannot be authenticated (e.g., a user attempting to send a data packet through the TOE that does not require authentication; in which case the identity is presumed to be authentic). In other cases (e.g., administrators, and authorized IT entities), the identity of the user is authenticated. It is impractical to require authentication of all users that attempt to send data through the TOE, therefore, the requirements specified in the TOE require authentication where it is deemed necessary. This does impose some risk that a data packet was sent from an identity other than specified in the data packet. FIA_ATD.1 defines the attributes of users, including a userid that is used to 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). This requirement allows a human user to have more than one user identity assigned, so that a single human user could assume all the roles necessary to manage the TOE. In order to ensure a separation of roles, this PP requires a single role to be associated with a user id. This is inconvenient in that the administrator would be required to log in with a different user id each time they wish to assume a different role, but this helps mitigate the risk that could occur if an administrator were to execute malicious code.

Objective	Requirements Addressing the Objective	Rationale
		FIA_UAU.1 contributes to this objective by limiting the services that are provided by the TOE to unauthenticated users. Management requirements and the unauthenticated information flow policy requirement provide additional control on these services. FIA_UAU_EXP.2 requires that administrators and authorized IT entities authenticate themselves to the TOE before performing administrative duties (including those performed by authorized IT entities (e.g., NTP server)), In order to control logical access to the TOE an authentication mechanism is required. The explicit requirement FIA_UAU_EXP.5 mandates that the TOE provide a local authentication mechanism. This requirement also affords the ST author the opportunity to add additional authentication mechanisms (e.g., single-use, certificates) if they desire.
		Local authentication is required to ensure someone that has physical access to the TOE and has not been granted logical access (e.g., a janitor) cannot gain unauthorized logical access to the TOE. The AVA_SOF.1 requirement is applied to the local 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. FTA_TSE.1.1 contributes to this objective by limiting a user's ability to logically access the TOE. This requirement provides the Security Administrator the ability to control when (e.g., time and day(s) of the week) and where (e.g., from a specific network address) remote administrators, as well as authorized IT entities can access the TOE. FIA_AFL.1-NIAP-0425 provides a detection mechanism for unsuccessful authentication attempts by remote administrators and authorized IT entities. The requirement enables a Security Administrator settable threshold that prevents unauthorized users from gaining access to authorized user's account by guessing authentication data by locking the targeted account until the Security Administrator takes some action (e.g., re-enables the account) or for some Security Administrator defined time period. Thus, limiting an unauthorized user's ability to gain unauthorized access to the TOE. The FTA_SSL family partially satisfies the O.ROBUST_TOE_ACCESS objective by ensuring that user's sessions are afforded some level of protection. FTA_SSL.1 provides the Security Administrator the capability to specify a time interval of inactivity in which an unattended local administrative session would be locked and will require the administrator responsible for that session to re-authenticate before the session can be used to access TOE resources. FTA_SSL.2 provides administrators the ability to lock their local administrative session. This component allows administrators to protect their session immediately, rather than waiting for the time-out period and minimizes their session's risk of exposure. FTA_SSL.3 takes into account remote sessions. After a Security Administrator defined time interval of inactivity remote sessions

Objective	Requirements Addressing the Objective	Rationale
		will be terminated, this refers to remote administrative sessions. This component is especially necessary, since remote sessions are not typically afforded the same physical protections that local sessions are provided.
O.TRUSTED_PATH The TOE will provide a means to ensure users are not communicating with some other entity pretending to be the TOE, and that the TOE is communicating with an authorized IT entity and not some other entity pretending to be an authorized IT entity.	FTP_ITC.1(1), FTP_ITC.1(2) FTP_TRP.1(1), FTP_TRP.1(2)	FTP_TRP.1.1 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, by requiring that the means used for invoking the communication path cannot be intercepted and allow a "man-in-the-middle-attack" (this does not prevent someone from capturing the traffic and replaying it at a later time – see FPT_RPL.1). Since the user invokes the trusted path (FTP_TRP.1.2) mechanism they can be assured they are communicating with the TOE. FTP_TRP.1.3 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. Furthermore, the remote administrator's communication path is encrypted during the entire session. FTP_ITC.1(1) and FTP_ITC.1(2) are similar to FTP_TRP.1(1) and FTP_TRP.1(2), in that they require a mechanism that creates a distinct communication path with the same characteristics, however FTP_ITC.1(1) and FTP_ITC.1(2) is used to protect communications between IT entities, rather than between a human user and an IT entity. FTP_ITC.1.3 requires the TOE to initiate the trusted channel, which ensures that the TOE has established a communication path with an authorized IT entity and not some other entity pretending to be an authorized IT entity.
O.VULNERABILITY_ANALYSIS_TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	AVA_VLA.3	To maintain consistency with the overall assurance goals of this TOE, O.VULNERABILITY_ANALYSIS_TEST 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. 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.

Table 6 - Rationale for TOE Security Requirements

6.4 Rationale for Assurance Requirements

The EAL definitions and assurance requirements in Part 3 of the CC were reviewed and the *Medium Robustness Assurance Package* as defined in Section 5.3 was believed to best achieve the goal of addressing circumstances where developers and

users require a moderate to high level of independently assured security in commercial products. The assurance package selection was based on:

- recommendations documented in the GIG;
- DoD Instruction 8500.1; and
- the postulated threat environment.

This collection of assurance requirements require TOE developers to gain assurance from good software engineering development practices which, though rigorous, do not require substantial specialist knowledge, skills, and other resources. Rationale for individual assurance requirements is provided in Table 6.

The Government's guidance in the GIG was consulted and found to also support the chosen assurance package. Specifically, the GIG states that medium robustness security services and mechanisms provide for additional safeguards above the DoD minimum and require good assurance security design as specified in EAL3 or greater.

The postulated threat environment specified in Section 3 of this PP was used in conjunction with the Information Assurance Technical Framework (IATF) Robustness Strategy guidance to derive the chosen assurance level.

These three factors were taken into consideration and the conclusion was that the medium robustness assurance package was the appropriate level of assurance.

6.5 Rationale for Not Satisfying All Dependencies

Each functional requirement, including explicit requirements was analyzed to determine that all dependencies were satisfied. All requirements were then analyzed to determine that no additional dependencies were introduced as a result of completing each operation. Table 7 identifies the functional requirement, its correspondent dependency and the analysis and rationale for not supporting the dependency in this PP.

Requirement	Dependency	Dependency Analysis and Rationale
FCS_CKM.1	FCS_CKM.2	The explicit requirement FCS_CKM_SYM_EXP.1 AND FCS_CKM_ASYM_EXP.1were chosen instead of FCS_CKM.2 to more clearly state the requirements as they apply to FIPS 140-2. Therefore, FCS_CKM_SYM_EXP.1 AND FCS_CKM_ASYM_EXP.1satisfies the

Requirement	Dependency	Dependency Analysis and Rationale
		dependency.
FCS_CKM.1 FCS_CKM.4	FMT_MSA.2	This dependency is satisfied by placing strict requirements on the values of attributes of the cryptographic module in the associated FCS requirements. Therefore, FMT_MSA.2 is not necessary to satisfy the requirement of only secure values being assigned to secure attributes.
FMT_MOF.1 FMT_MSA.1 FMT_MTD.1	FMT_SMF.1	The requirements FMT_MOF.1, FMT_MSA.1, FMT_MTD.1 express the functionality required by the TSF to provide the specified functions to manage TSF data, security attributes, and management functions. These requirements make clear that the TSF has to provide the functions to manage the identified data, attributes, and functions. Therefore, FMT_SMF.1 is not necessary.
FIA_UAU.1 FMT_SMR.2	FIA_UID.1	This dependency is satisfied with the inclusion of requirement FIA_UID.2. This requirement is hierarchical to FIA_UID.1 and is sufficient to satisfy the dependency for these requirements.
FMT_MOF.1 FMT_MSA.1 FMT_MTD.2 FMT_REV.1	FMT_SMR.1	This dependency is satisfied with the inclusion of requirement FMT_SMR.2. This requirement is hierarchical to FMT_SMR.1 and is sufficient to satisfy the dependency for these requirements.

Table 7 - Unsupported Dependency Rationale

6.6 Rationale for Strength of Function Claim

Part 1 of the CC defines "strength of function" in terms of the minimum efforts assumed necessary to defeat the expected security behavior of a TOE security function. There are three strength of function levels defined in Part 1: SOF-basic, SOF-medium and SOF-high. SOF-medium is the strength of function level chosen for this PP. SOF-medium states, "a level of the TOE strength of function where analysis shows that the function provides adequate protection against straightforward or intentional breach of TOE security by attackers possessing a moderate attack potential." The rationale for choosing SOF-medium was to be consistent with the TOE objective O.VULNERABILITY_ANALYSIS_TEST and assurance requirements included in this PP. Specifically, AVA_VLA.3 requires that the TOE be resistant to an attacker with a moderate-attack potential, this is consistent with SOF-medium. Consequently, the metrics (i.e., passwords and keys) chosen for inclusion in this PP were determined to be acceptable for SOF-medium and would adequately protect information in a Medium Robustness Environment.

6.7 Rationale for Explicit requirements

Table 8 presents the rationale for the inclusion of the explicit functional and assurance requirements found in this PP. The explicit requirements that are included as NIAP interpretations do not require a rationale for their inclusion per CCEVS management.

Explicit Requirement	Identifier	Rationale
FAU_ARP_ACK_EXP.1	Security alarm acknowledgement	This explicit requirement is necessary since a CC requirement does not exist to ensure an administrator will be aware of the alarm. The intent is to ensure that if an administrator is logged in and not physically at the console or remote workstation the message will remain displayed until the administrators have acknowledged it. The message will not be scrolled off the screen due to other activity-taking place (e.g., the auditor is running an audit report).
FCS_BCM_EXP.1	Baseline cryptographic module	This explicit requirement is necessary since the CC does not provide a means to specify a cryptographic baseline of implementation.

Explicit Requirement	Identifier	Rationale
FCS_CKM_SYM_EXP.1	Cryptographic Key Establishment for AES symmetric keys	This two explicit requirements are necessary since the CC does not specifically address concepts of key distribution and the nature of the requirements as specified by FIPS 140-2.
FCS_CKM_ASYM_EXP.1	Cryptographic Key Entry for Digital Signature/verification private keys	
FCS_COP.EXP.2	Cryptographic operation (Encryption/Decryption using AES)	This explicit requirement is necessary since the CC does not specifically provide for specifying modes of operation and was necessary to accommodate the FIPS 140-2 aspects.
FCS_COP.EXP.3	Cryptographic operation (Digital Signature Generation/Verification)	This explicit requirement is necessary since the CC does not specifically provide for the specific aspects that are required by FIPS 140-2 with respect to digital signatures.
FCS_COP.EXP.5	Cryptographic operation (Random number generation	This explicit requirement is necessary since the CC cryptographic operation components are focused on specific algorithm types and operations requiring specific key sizes.
FCS_COP.EXP.6	Cryptographic operation (Cryptographic Hashing Function)	This explicit requirement is necessary since the CC does not specifically provide for the specific aspects that are required by FIPS 140-2 with respect to hashing functions.
FIA_UAU_EXP.5	Authentication mechanism	This explicit requirement is needed for authentication because there is no CC requirement that explicitly requires the TSF provide authentication. This requirement also serves as a place for ST authors to add additional authentication mechanisms if they desire.
FPT_TST_EXP.4	TSF testing (with cryptographic integrity verification)	This explicit requirement is necessary to capture the notion of the TOE using cryptography to verify the integrity of the TSF software. Additionally, the TSF data set that is subject to these tests was reduced to address the notion that it does not make sense to test the integrity of some TSF data (e.g., audit data) and this explicit requirement address that.

Explicit Requirement	Identifier	Rationale
FPT_TST_EXP.5	Cryptographic self-test	The PP authors felt that the TSF self tests did not adequately address the notion of testing certain aspects of the TSF upon the completion of an operation. This explicit requirement is necessary to capture the notion of the TOE having the ability to test the cryptographic components immediately after the generation of a key. The CC does not contain a requirement that addresses this notion.
ADV_ARC_EXP.1	Architectural Description	These explicit assurance requirements were deemed necessary by NSA to reduce the ambiguity in the associated CC assurance families and to provide the level of assurance appropriate for medium robustness environments.
ADV_FSP_EXP.1	Functional Specification with Complete Summary	
ADV_HLD_EXP.1	Security-Enforcing High-Level Design	
ADV_INT_EXP.1	Modular Decomposition	
ADV_LLD_EXP.1	Security-Enforcing Low-Level Design	
AVA_CCA_EXP.2	Systematic Cryptographic Module Covert Channel Analysis	

Table 8 - Rationale for Explicit Requirements

7.0 ADV EXPLICIT ASSURANCE BACKGROUND INFORMATION

7.1 ADV_INT_EXP

This explicit component was created to levy different modularity metrics on the SFP-enforcing modules and non-SFP-enforcing modules.

The parts of the TSF that implement an SFP (in this component, SFP-enforcing is used to designate modules that enforce an SFP) that is determined and assigned by the PP/ST author, are those modules that interact (defined in the coupling analysis) with the module or modules that provide the TSFI for that SFP with justified exceptions. The intent is that all of the modules that play an SFR related role (as opposed to modules that provide infrastructure support, such as scheduling, reading binary data from the disk) in enforcing an SFP are identified as SFP-enforcing. The remaining modules in the TSF are deemed non-SFP-enforcing modules, since they could be TSP-enforcing (e.g., enforcing a policy not assigned to this component), as well as TSP-supporting.

Objectives

This component addresses the internal structure of the software TSF. The SFP-enforcing modules require stricter adherence to the coupling and cohesion metrics than the metrics levied on the non-SFP-enforcing modules due to their key role in policy enforcement. While the non-SFP-enforcing modules also play a role in enforcing policy, 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.

Requirements are presented for modular decomposition of the SFP-enforcing and non-SFP-enforcing functionality within the TSF. These requirements, when applied to the internal structure of the TSF, should result in improvements that aid both the developer and the evaluator in understanding the TSF, and also provides the basis for designing and evaluating test suites. Further, improving understandability of the TSF should assist the developer in simplifying its maintainability. The principal goal achieved by inclusion of the requirements from the ADV_INT class in a PP/ST is understandability of the TSF.

Modular design aids in achieving understandability by clarifying what 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 illuminating the design of a module by using 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. Additional desirable properties of modular decomposition are a reduction in the amount of redundant or unneeded code.

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 subtlety or complexity in a module's design. These deviations from the ideals of modular decomposition are often deemed necessary to achieve some goal or constraint, be it related to performance, compatibility, future planned functionality, or some 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.

Another 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. For example, coding standards ensure that meaningful names are given to variables and data structures, the code has a structure that is similar to code developed by other programmers, loops used in the code are understandable (e.g., leaving a loop to another section of code and returning is undesirable), 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 maintenance, and assists the development team in performing code walk-throughs. Some aspects of coding standards are specific to a given program language (e.g., the C language may have a different standard than the Java language or assembly level code). It is expected that the coding standards are appropriately followed for the employed programming language(s). 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.

Design complexity minimization is a key characteristic of a reference validation mechanism, the purpose of which is to arrive at a TSF that is easily understood so that it can be completely analyzed. (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.)

Application notes

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.

This component requires the PP or ST author to fill in an assignment with the SFPs that are felt to be critical to the TOE and therefore their resulting design and implementation require stricter metrics for modularity. The SFPs can be those explicitly identified in the CC (i.e., FDP_ACC, FDP_IFF) by simply placing the appropriate label as specified in those requirements, or other policies determined by the PP/ST author (e.g., I&A, Audit), in which case, the PP/ST author should explicitly identify all of the SFRs that they intend to satisfy a policy that is not explicitly stated in the CC. This is necessary since currently a convention does not exist to place a convenient label on these policies.

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 as depicted in the Figure **AA**, where is this example, non-SFP-enforcing is everything in the TSF other than the SFP-enforcing functions.

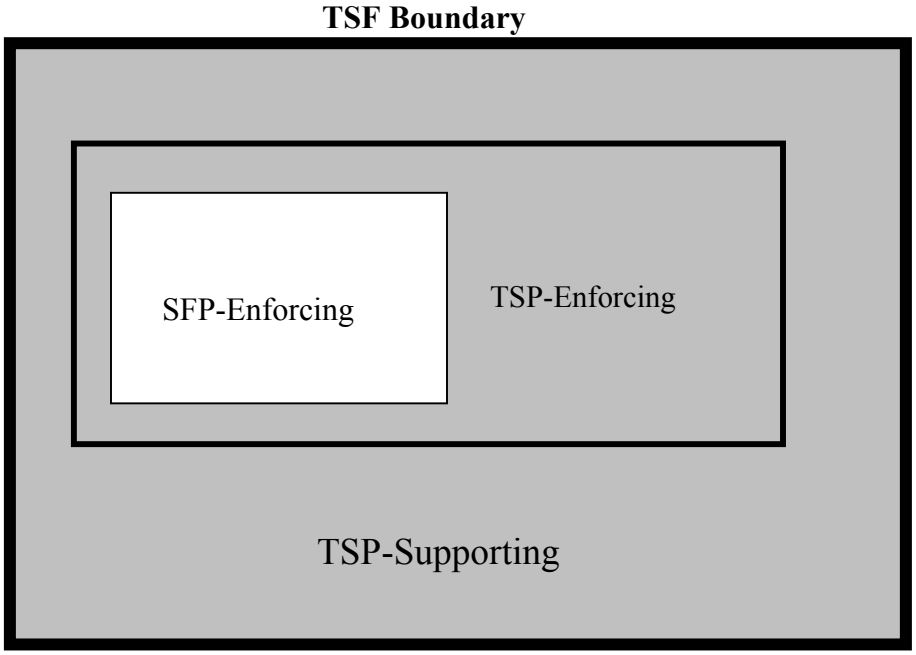


Figure AA. SFP-enforcing may only be a subset of TSP-enforcing functions.

The developer is required to identify the modules that are SFP-enforcing and implicitly the remaining modules, which will be non-SFP-enforcing. As stated earlier, the SFP-enforcing modules are those modules that interact with the module or modules that provide the TSFI for that SFP with justified exceptions. The justification of the non-SFP-enforcing modules (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. As depicted in the Figure **XX** below, if a TSFI has already been designated as non-SFP-enforcing then the designation of the modules interacting with the module providing the TSFI do not have to be justified (e.g., modules X, Y,

Z). The justification of the designation is only necessary for the module(s) that interact with a module that provides a TSFI that is SFP-enforcing (e.g., modules D, E, F (since it is writing to a global variable that Module A is reading, but in this example, it is not an SFP-enforcing variable).

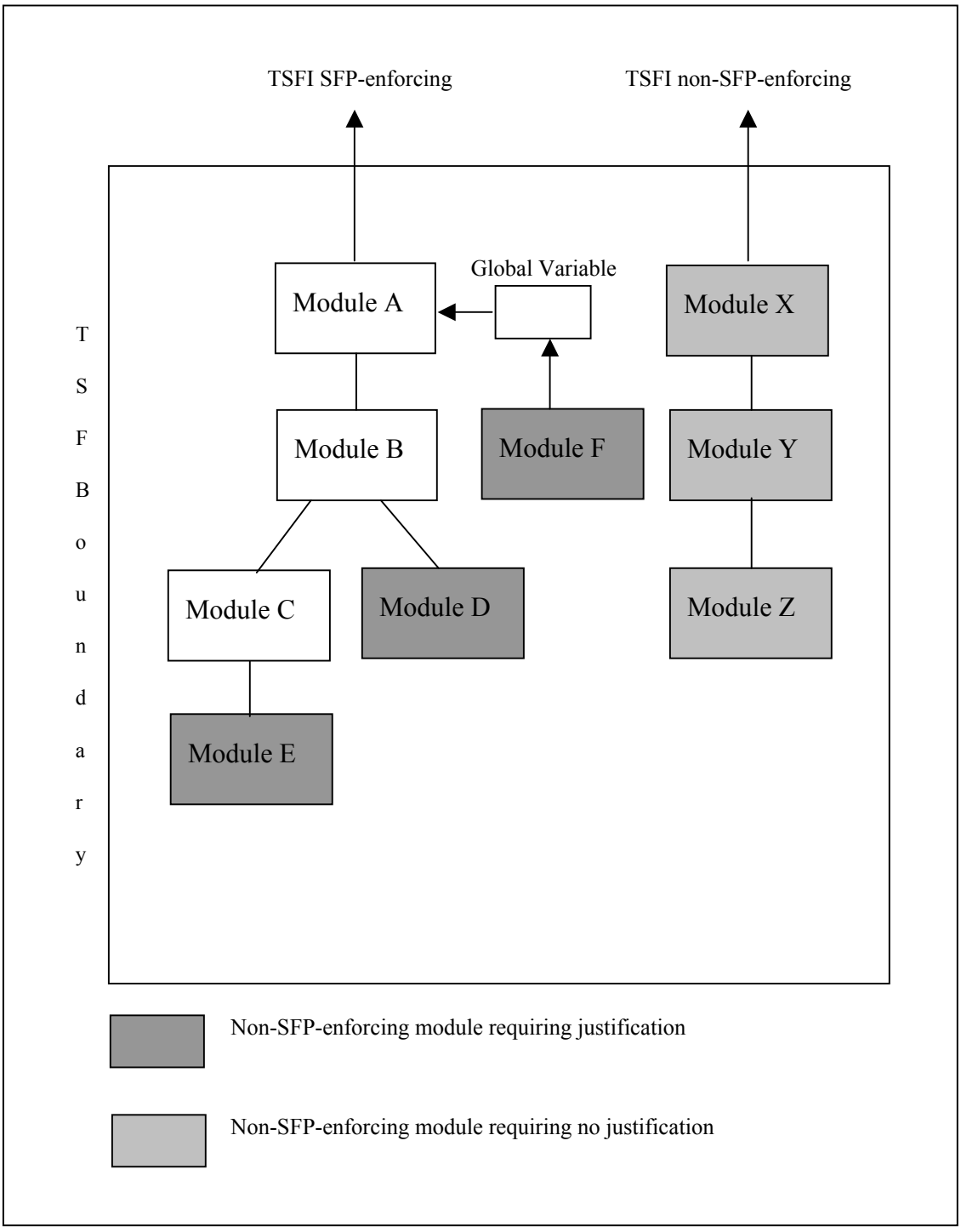


Figure XX. Example of non-SFP-enforcing modules requiring justification.

The modules identified in the architectural description are the same as the modules identified in the low-level design.

Terms, definitions and background

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.[4]

- Common coupling through global variables is generally allowed, but only 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

---

[4] *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.*

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 effectively 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.

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 generating code that is more readily understood.

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.

7.2 ADV_FSP_EXP.1

Objectives

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 has to completely address all of the user-visible TOE security functional requirements.

Application notes

A description of the TSF interfaces (TSFI) provides fundamental evidence on which assurance in the TOE can be built. Fundamentally, 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.

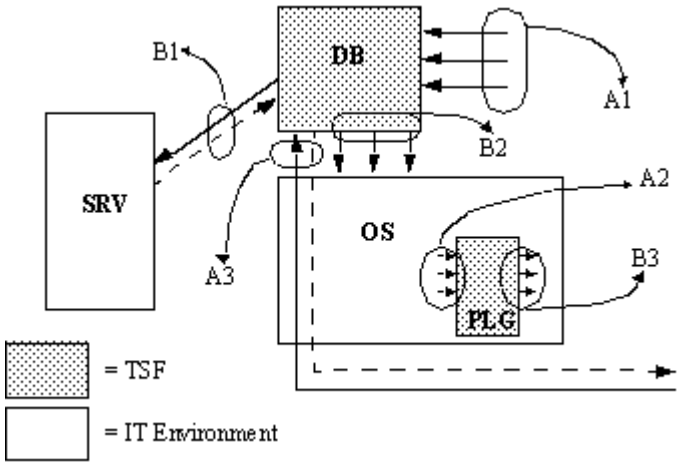
In order to identify the software interfaces to the TSF, the parts of the TOE that make up the TSF must be identified. This identification is formally a part of ADV_HLD_EXP analysis. In this analysis, a portion of the TOE is considered to be in the TSF under two conditions:

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.

Identification of the TSFI is a complex undertaking. The TSF is providing services and resources, and so the TSFI are interfaces *to* the security services/resources the TSF is providing. This is especially relevant for TSFs that have dependencies on the IT environment, because not only is the TSF providing security services (and thus exposing TSFI), but it is also *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.

This concept (and concepts to be discussed in the following paragraphs) is illustrated in the following figure.



The figure above illustrates a TOE (a database management system) that has dependencies on the IT environment. The shaded boxes represent the TSF, while the unshaded boxes represent IT entities in the environment. The TSF comprises the database engine and management GUIs (represented by the box labeled "DB") and a kernel module that runs as part of the OS that performs some security function (represented by the box labeled "PLG"). The TSF kernel module has entry points defined by the OS specification that the OS will call to invoke some function (this could be a device driver, or an authentication module, etc.). The key is that this pluggable kernel module is providing security services specified by functional requirements in the ST. The IT environment consists of the operating system (represented by the box labeled "OS") itself, as well as an external server (labeled SRV). This external server, like the OS, provides a service that the TSF depends on, and thus needs to be in the IT environment. Interfaces in the figure are labeled Ax for TSFI, and Bx for interfaces to be documented in AGD_ING. Each of these groups of interfaces is now discussed.

Interface group A1 represent the prototypical set of TSFI. These are interfaces used to directly access the database and its security functionality and resources.

Interface group A2 represent the TSFI that the OS invokes to obtain the functionality provided by the pluggable module. These are contrasted with interface group B3, which represent calls that the pluggable module makes to obtain services from the IT environment.

Interface group A3 represent TSFI that "pass through" the IT environment. In this case, the DBMS communicates over the network using a proprietary application-level protocol. While the IT environment is responsible for providing various supporting protocols (e.g., Ethernet, IP, TCP), the application layer protocol that is used to obtain services from the DBMS is a TSFI and must be documented as such. The dotted line indicates return values/services from the TSF over the network connection.

Non-TSFI interfaces pictured are labeled Bx. Interface group B1 is the most complex of these, because the architecture of the system and environmental assumptions and conditions will drive its analysis. In the first case, assume that, either through an environmental assumption or an IT environmental requirement, the network link between the DB and SRV is protected (it could be on a separate subnet, or it could be protected by a firewall such that only the DB could connect to the port on the SRV) such that only the DB has access to the SRV. In this case, the interface needs only to be documented in the integrator guidance, since untrusted users are unable to gain access.

However, consider the case where SRV is now just "somewhere on the network", and now the port that the DB opens up to communicate with the SRV is "exposed" to untrusted users. In this case, while the interface presented by the DB (the TSF) still only needs to be documented in the integrator guidance, additional considerations with respect to vulnerabilities may need to be documented as part of the AVA_VLA activity because of this exposure.

In the course of performing its functions, the DB will make system calls down to the OS. This is represented by interface group B2. While these calls are not part of the TSFI, they are an interface that needs to be documented in the integrator guidance.

Interface group B3, mentioned previously in connection with interface group A2, is similar to interface group B2 in that these are calls made by the TSF to the IT environment to perform services for the TSF.

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.

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.

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. Other interfaces supported by the tool that the administrator is directed not to use (and thus play no role in supporting the TSP), but that are accessible to non-administrators, are not part of the TSFI because there are no privileges associated with their use. 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.

It is also important to note that some TOEs will have interfaces that one might consider part of the TSFI, but environmental factors remove them from consideration (an example is the case of interface group B1 discussed earlier). Most of these examples are for TOEs to which untrusted users have restricted access. For example, consider a firewall that untrusted users only have access to via the network interfaces, and further that the network interfaces available only support packet-passing (no remote administration). Further suppose that the firewall had a command-line interface that logged-in administrators could use to administer the system, or they could use a GUI-based tool that essentially translated the GUI-based checkboxes, textboxes, etc., into scripts that invoked the command-line utilities. Finally, suppose that the administrators were directed in the administrative guidance to use the GUI-based tool in administering the firewall. In this case, the command-line interface does not have to be documented because it is inaccessible to untrusted users, and because the administrators are instructed not use it.

The term "administrator" above is used in the sense of an entity that has complete trust with respect to all policies implemented by the TSF. There may be entities that are trusted with respect to some policies (e.g., audit) and not to others (e.g., a flow control policy). In these cases, even though the entity may be referred to as an "administrator", they need to be treated as untrusted users with respect to policies to which they have no administrative access. So, in the previous firewall example, if there was an auditor role that was allowed direct log-on to the firewall machine, the command-line interfaces not related to audit are now part of the TSFI, because they are accessible to a user that is not trusted with respect to the policies the interfaces provide access to. The point is that such interfaces need to be addressed in the same manner as previously discussed.

Hardware interfaces exist as well. Functions provided by the BIOS of various devices may be visible through a "wrapper" interface such as the IOCTLs in a Unix operating system. If the TOE is or includes a hardware device (e.g., a network interface card), the bus interface signals, as well as the interface seen at the network port, must be considered "interfaces." Switches that can change the behavior of the hardware are also part of the interface.

As indicated above, an interface exists at the TSF boundary if it can be used (by an administrator; untrusted user; or another TOE) to affect the behavior of the TSF. The requirements in this family apply to all types of TSFI, not just APIs.

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.

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.

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).

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.

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.

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.

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.

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).

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.

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).

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.

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.[5] 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.

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.

[5] *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."*

7.3 ADV_HLD_EXP.1

Objectives

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.

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 should be able to distinguish between the subsystems that 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.

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 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

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".

A security enforcing 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.

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.

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 1) all security-enforcing behavior has been identified and 2) the subsystems or portions of subsystems that are security supporting have been correctly classified.

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, 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 high-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.

7.4 ADV_LLD_EXP.1

Objectives

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

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.

A security-enforcing module is a module that directly implements a security-enforcing TSFI. While this could, for example, include all modules in the call-tree of a security-enforcing module, typically there will be some modules in the call-tree of a security-enforcing module that are not themselves security enforcing. 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.

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 to the implementation of the module. For instance, the low-level design may describe a block of processing that is looped over a number of times. The actual implementation may be a *for* loop or a *do* loop, both of which could be used to implement the algorithm. Likewise, a collection of objects could be represented by a linked list or an array; this level of detail is not required to be presented, since both are algorithmically identical. Conversely, if a module's actual implementation performed a bubble sort, it would be inadequate for the low-level design to specify that the module "performed a sort"; it would have to describe the type of sort that was being performed.

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.

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; 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.

The interfaces presented by a module are those interfaces used by other modules to invoke the functionality provided. Interfaces are described in terms of how 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.

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 algorithmic reason the invoking module is being called. For instance, if Module A is being described, and it uses Module B's bubble sort routine, an inadequate algorithmic description would be "Module A invokes the double_bubble() interface in Module B to perform a bubble sort." An adequate algorithmic description would be "Module A invokes the double_bubble routine with the list of access control entries; double_bubble() will return the entries sorted first on the username, then on the access_allowed field according the following rules..." The low-level design must provide enough detail so that it is clear what effects Module A is expecting from the bubble sort interface. Note that one method of presenting these called interfaces is via a call tree, and then the algorithmic description can be included in the algorithmic description of the called module.

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.

The purpose a module fulfills is a short description indicating what function the module provides. The level of detail provided should be such that the reader could get a general idea of what the module's function is in the architecture, and to determine (for security-supporting modules) that it is not a security-enforcing module.

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.

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.

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.

7.5 ADV_ARC_EXP.1

Objectives

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

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.

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.

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. However, FPT_RVM is concerned with interfaces that 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.

For FPT_SEP, the design description should cover how user input is handled by privileged-mode routine; 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.

For FPT_RVM, the description should cover resources that are protected under the SFRs (usually FDP_* components) and 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.

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.

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.

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.

8.0 REFERENCES

1) Common Criteria for Information Technology Security Evaluation*, CCIB-98-031 Version 2.1, August 1999.*

2) *Department of Defense Chief Information Officer Guidance and Policy Memorandum No. 6-8510,* Guidance and Policy for the Department of Defense Global Information Grid Information Assurance (GIG), *June 2000.*

3) U.S. Government Traffic-Filter Firewall Protection Profile for Medium Robustness Environments, *Version 1.4, May 1, 2000.*

4) U.S. Department of Defense Application-Level Firewall Protection Profile for Medium Robustness Environments, *Version 1.1, December 2001.*

5) Information Assurance Technical Framework, *Version 3.0, September 2000.*

6) *Federal Information Processing Standard Publication (FIPS-PUB) 46-3,* Data Encryption Standard (DES), *October 1999.*

7) *Federal Information Processing Standard Publication (FIPS-PUB) 140-2,* Security Requirements for Cryptographic Modules, *May 25, 2001.*

8) *Internet Engineering Task Force,* IP Encapsulating Security Payload (ESP), *RFC 2406, November 1998.*

9) *Internet Engineering Task Force,* Internet Key Exchange (IKE), *RFC 2409, November 1998.*

10) *Internet Engineering Task Force,* ESP CBC-Mode Cipher Algorithms, *RFC 2451, November 1998.*

11) *Internet Engineering Task Force,* Use of HMAC-SHA-1-96 within ESP and AH*, RFC 2404, November 1998.*

12) *Department of Defense Directive, Information Assurance*, 8500.1, October 24, 2002.

13) *Department of Defense Instruction, Information Assurance Implementation*, 8500.2, February 6, 2003.

14) The AES Cipher Algorithm and Its Use with IPsec <draft-ietf-ipsec-ciph-aes-cbc.03.txt>, *Internet draft, November 2001.*

15) *Federal Information Processing Standard Publication (FIPS-PUB) 197,* Specification for the Advanced Encryption Standard (AES)*, November 26, 2001*

9.0 TERMINOLOGY

In the Common Criteria, many terms are defined in Section 2.3 of Part 1. The following are a definitions of terms used in this PP and common to other DoD PPs.

Access -- Interaction between an entity and an object that results in the flow or modification of data.

Access Control -- Security service that controls the use of resources[6] and the disclosure and modification of data.[7]

Accountability -- Property that allows activities in an IT system to be traced to the entity responsible for the activity.

Administrator -- A user who has been specifically granted the authority to manage some portion or all of the TOE and 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.

---

[6] Hardware and software.

[7] Stored or communicated.

Availability -- Timely[8], reliable access to IT resources.

Compromise -- Violation of a security policy.

Confidentiality -- A security policy pertaining to disclosure of data.

Critical Security Parameters (CSP) -- Security-related information (e.g., cryptographic keys, authentication data such as passwords and pins, and 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 [7]:

- the transformation of plaintext data into ciphertext data,
- the transformation of cipher text 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, firmware, or some combination thereof that implements cryptographic logic or processes, including cryptographic algorithms, and is contained within the cryptographic boundary of the module.

Cryptographic Module Security Policy -- A precise specification of the security rules under which a cryptographic module must operate, including the rules derived from the requirements of this PP and additional rules imposed by the vendor.

Defense-in-Depth (DID) -- 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/or groups to which they belong. These controls are discretionary in the

---

[8] According to a defined metric.

sense that a subject with certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject.

DMZ -- A Demilitarized Zone (DMZ) is a network that is mediated by the TOE but, as a result of less stringent access controls, provides access to publicly available services, such as web servers.

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 may be based on physical location and proximity.

Entity -- A subject, object, user or another IT device, which interacts with TOE objects, data, or resources.

External IT entity -- Any trusted Information Technology (IT) product or system, outside of the TOE, which may, in accordance with the TSP, perform an operation.

Identity -- A representation (e.g., a string) uniquely identifying an authorized user, which can either be the full or abbreviated name of that user or a pseudonym.

Integrity -- A security policy pertaining to the corruption of data and TSF mechanisms.

Integrity label -- A security attribute that represents the integrity level of a subject or an object. Integrity labels are used by the TOE as the basis for mandatory integrity control decisions.

Integrity level -- The combination of a hierarchical level and an optional set of non-hierarchical categories that represent the integrity of data.

Mandatory Access Control (MAC) -- A means of restricting access to objects based on subject and object sensitivity labels.[9]

Mandatory Integrity Control (MIC) -- A means of restricting access to objects based on subject and object integrity labels.

Multilevel -- The ability to simultaneously handle (e.g., share, process) multiple levels of data, while allowing users at different sensitivity levels to access the system concurrently. The system permits each user to access only the data to which they are authorized access.

Named Object[10] -- An object that exhibits all of the following characteristics:

---

[9] The Bell LaPadula model is an example of Mandatory Access Control

[10]The only named objects in this PP, are operating system controlled files.

- 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.

(Note: Due to the deletion of the last sentence in the OS PP (pertaining to intended use of the object being for sharing user data), something may need to be done to the requirements section of the PP (i.e., FDP_ACF) to ensure that some objects, which may satisfy the above but which are not intended for sharing user data do not need a full DAC implementation but rather it is acceptable if they are "owner only" or some other appropriate mechanism.)

Non-Repudiation -- A security policy pertaining to providing one or more of the following:

- To the sender of data, proof of delivery to the intended recipient,

- To the recipient of data, proof of the identity of the user who sent the data.

Object -- An entity within the TSC 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.

Operating System (OS) -- An 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.

Operational key -- Key intended for protection of operational information or for the production or secure electrical transmissions of key streams.

Peer TOEs -- Mutually authenticated TOEs that interact to enforce a common security policy.

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.

Robustness -- A characterization of the strength of a security function, mechanism, service or solution, and the assurance (or confidence) that it is implemented and functioning correctly. DoD has three levels of robustness:

- **Basic:** Security services and mechanisms that equate to good commercial practices.

- **Medium:** Security services and mechanisms that provide for layering of additional safeguards above good commercial practices.

- **High:** Security services and mechanisms that provide the most stringent protection and rigorous security countermeasures.

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 level -- The combination of a hierarchical classification and a set of non-hierarchical categories that represent the sensitivity on the information [10].

Sensitivity label -- A security attribute that represents the security level of an object and that describes the sensitivity (e.g. Classification) of the data in the object. Sensitivity labels are used by the TOE as the basis for mandatory access control decisions [10].

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 entity within the TSC that causes operations to be performed.

Symmetric key -- A single, secret key used for both encryption and decryption in symmetric cryptographic algorithms.

Threat -- Capabilities, intentions and attack methods of adversaries, or any circumstance or event, with the potential to violate the TOE security policy.

Threat Agent - Any human user or Information Technology (IT) product or system which may attempt to violate the TSP and perform an unauthorized operation with the TOE.

User -- Any entity (human user or external IT entity) outside the TOE that interacts with the TOE.

Vulnerability -- A weakness that can be exploited to violate the TOE security policy.

10.0 ACRONYMS

The following abbreviations from the Common Criteria are used in this Protection Profile:

AES	Advanced Encryption Standard
ATM	Asynchronous Transfer Method
CC	Common Criteria for Information Technology Security Evaluation
DES	Data Encryption Standard
DoD	Department of Defense
DMZ	Demilitarized zone
EAL	Evaluation Assurance Level
ESP	Encapsulating Security Payload
FIPS PUB	Federal Information Processing Standard Publication
GIG	Global Information Grid
I&A	Identification and Authentication
IATF	Information Assurance Technical Framework
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IPSEC ESP	Internet Protocol Security Encapsulating Security Payload
IP	Internet Protocol
IT	Information Technology
MRE	Medium Robustness Environment
NBIAT&S	Network Boundary Information Assurance Technologies and Solutions Support
NIAP	National Information Assurance Partnership
NIST	National Institute of Standards and Technology
NSA	National Security Agency
NTP	Network Time Protocol
PKI	Public Key Infrastructure
PP	Protection Profile
RNG	Random Number Generator
SFP	Security Function Policy
SMTP	Simple Mail Transfer Protocol
SOF	Strength of Function
ST	Security Target

TOE	Target of Evaluation
TSE	TOE Security Environment
TSF	TOE Security Function
TSP	TOE Security Policy
UDP	User Datagram Protocol
URL	Uniform Research Locator
VPN	Virtual Private Network

11.0 REFINEMENTS

This section contains refinements where text was omitted[11]. Omitted text is shown as bold text within parenthesis. The actual text of the functional requirements as presented in Section 5 has been retained.

FAU_ARP.1.1 – **Refinement:** The TSF shall (**take)** [immediately display an alarm message, identifying the potential security violation and make accessible the audit record contents associated with the auditable event(s) that generated the alarm, at the:

- local console,
- remote administrator sessions that exist, and;
- remote administrator sessions that are initiated before the alarm has been acknowledged, and;
- at the option of the Security Administrator, generate an audible alarm, and;
- [assignment: other methods]] upon detection of a potential security violation.

FAU_GEN.1.1-NIAP-0410 – **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) All auditable events (**for the) [**listed in Table 3] (**level of audit; and)**

c) [selection: [assignment: events at a basic level of audit introduced by the inclusion of additional SFRs determined by the ST Author], [assignment: events commensurate with a basic level of audit introduced by the inclusion of explicit requirements determined by the ST Author], no additional events].

FAU_GEN.1.2-NIAP-0410 – **Refinement**: 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 (if applicable), and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [(**selection)**: [information specified in column three of Table 3 below].

FAU_GEN.2.1-NIAP-0410 – **Refinement**: (**For audit events resulting from actions of identified users**,) the TSF shall be able to associate each auditable event with the identity of the user that caused the event.

FAU_SAA.1.2-NIAP-0407 - Refinement: The TSF shall (**monitor) enforce the following rules for monitoring audited events**: a) accumulation or combination of [**the following events**:

a) Security Administrator specified number of authentication failures;

b) Security Administrator specified number of Information Flow policy violations by an individual presumed source network identifier (e.g., IP address) within an administrator specified time period;

c) Security Administrator specified number of Information Flow policy violations to an individual destination network identifier within an administrator specified time period;

d) Security Administrator specified Information Flow policy rule, or group of rule violations within an administrator specified time period;

e) Any detected replay of TSF data or security attributes;

f) Any failure of the cryptomodule self-tests (FPT_TST_EXP.5);

g) Any failure of the other TSF self-tests (FPT_TST_EXP.4);

h) Security Administrator specified number of encryption failures;

i) Security Administrator specified number of decryption failures] known to indicate a potential security violation;

j) [selection: [assignment: any other rules], "no additional rules"].

FAU_STG.3.1 - **Refinement**: The TSF shall (**take)** [immediately alert the administrators by displaying a message at the local console, and at the remote administrative console when an administrative session exists for each of the defined administrative roles, at the option of the Security Administrator generate an audible alarm, [selection: [assignment: other methods determined by the ST Author], no other methods] if the audit trail exceeds [a Security Administrator settable percentage of storage capacity].

FAU_STG.1.2-NIAP-0423 – **Refinement:** The TSF shall be able to *prevent* **(unauthorized)** modifications to the audit records in the audit trail.

FCS_CKM.1.1 **Refinement**: The (**TSF) cryptomodule** shall generate **symmetric** cryptographic keys (**in accordance with a specified cryptographic key generation algorithm)** [using a FIPS-Approved Random Number Generator**] (and specified cryptographic key sizes**) [for all key sizes] that meet the following: [one of the standards defined in Annex C to FIPS 140-2].

FCS_CKM.4.1 - **Refinement**: The TSF shall destroy cryptographic keys in accordance with a (**specified cryptographic key destruction method)** [cryptographic key zeroization method**]** that meets the following:

a) [The Key Zeroization Requirements in FIPS PUB 140-2 Key Management Security Levels 3;

b) Zeroization of all private cryptographic keys, plaintext cryptographic keys and all other critical cryptographic security parameters shall be immediate and complete; and

c) The zeroization shall be executed by overwriting the key/critical cryptographic security parameter storage area three or more times with an alternating pattern.

d) The TSF shall overwrite each intermediate storage area for private cryptographic keys, plaintext cryptographic keys, and all other critical security parameters three or more times with an alternating pattern upon the transfer of the key/CSPs to another location.]

FDP_IFF.1.2-NIAP-0417 **- Refinement:** The TSF shall permit an information flow between a **source** (**controlled)** subject and **a destination subject** (**controlled information)** via a controlled operation if the following rules hold:

- **[**the presumed identity of the source subject is in the set of source subject identifiers;
- the identity of the destination subject is in the set of source destination identifiers;
- the information security attributes match the attributes in an information flow policy rule (contained in the information flow policy ruleset defined by the Security Administrator) according to the following algorithm [assignment: algorithm used by the TOE to match information security attributes to information flow policy rules]; and
- the selected information flow policy rule specifies that the information flow is to be permitted].

FDP_IFF.1.2-NIAP-0417(2) – **Refinement**: The TSF shall permit an information flow between a **source** (**controlled)** subject and (**controlled information) the TOE** via a controlled operation if the following rules hold:

- [the presumed identity of the source subject is in the set of source subject identifiers;
- the identity of the destination subject is the TOE;

- the information security attributes match the attributes in an information flow control policy according to the following algorithm [assignment: algorithm used by the TOE to match information security attributes to information flow control policy].

FIA_AFL.1.2-NIAP-0425 – **Refinement:** When the defined number of unsuccessful authentication attempts has been met (**or surpassed)**, the TSF shall [at the option of the Security Administrator prevent the remote administrators from performing activities that require authentication until an action is taken by the Security Administrator, or until a Security Administrator defined time period has elapsed].

FMT_MSA.3.1-NIAP-0409(1) – **Refinement:** The TSF shall enforce the [UNAUTHENTICATED INFORMATION FLOW SFP, AUTHENTICATED INFORMATION FLOW SFP] to provide *restrictive* default values for (**security attributes) the information flow policy ruleset** that (**are**) is used to enforce the SFP.

FMT_MSA.3.1-NIAP-0409(2) – **Refinement:** The TSF shall enforce the [UNAUTHENTICATED TOE SERVICES SFP] to provide *restrictive* default values for (**security attributes) (that are used to enforce the SFP) the set of TOE services available to unauthenticated users.**

FMT_REV.1.2 - **Refinement:** The TSF shall **immediately** enforce the (**rules)**:

- [revocation of a user's role (Security Administrator, Cryptographic Administrator, Audit Administrator);
- changes to the information flow policy ruleset when applied;
- [selection: [assignment: other rules as determined by the ST Author], none]].

FPT_SEP.2.3 - **Refinement:** The TSF shall maintain the part of the TSF related to [cryptography] in (**security domain(s**)) **an address space** for (**their) its** own execution that protects (**them**) **it** from interference and tampering by the remainder of the TSF and by subjects untrusted with respect to (**those SFPs) the cryptographic functionality**.

FTA_SSL.1.2 - **Refinement:** The TSF shall require the **(following events to occur) user to re-authenticate** prior to unlocking the session(**: [assignment: events to occur]).**

FTA_SSL.2.2 - **Refinement:** The TSF shall require the (**following events to occur) user to re-authenticate** prior to unlocking the session(**: [assignment: events to occur]).**

FTA_TSE.1.1 - **Refinement:** The TSF shall be able to deny (**session)** establishment **of an authorized user session** based on [location, time, and day].

FTP_ITC.1.1(1) - **Refinement:** The TSF shall (**provide**) **use encryption to** provide a **trusted** communication channel between itself and (**a remote trusted IT product) authorized IT entities** that is logically distinct from other communication channels and

provides assured identification of its end points and protection of the channel data from (**modification or)** disclosure.

FTP_ITC.1.1(2) - **Refinement:** The TSF shall (**provide**) **use a cryptographic signature** to provide a **trusted** communication channel between itself and (**a remote trusted IT product) authorized IT entities** that is logically distinct from other communication channels and provides assured identification of its end points and (**protection of the channel data from modification or disclosure) detection of the modification of data**.

FTP_TRP.1.1(1) - **Refinement:** The TSF shall provide (**a**) **an encrypted** communication path between itself and *remote **administrators and authenticated proxy*** users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from **(modification or)** disclosure.

FTP_TRP.1.1(2) - **Refinement:** The TSF shall **use a cryptographic signature to** provide a communication path between itself and *remote **administrators and authenticated proxy*** users that is logically distinct from other communication paths and provides assured identification of its end points and (**protection**) **detection (of the communicated data from) modification (or disclosure) of data**.

FTP_ITC.1.1(1) - **Refinement:** The **(TSF) IT Environment** shall provide a **trusted** communication channel between itself and the **(a remote trusted IT product) TSF** that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from **(modification or)** disclosure.

FTP_ITC.1.1(2) - **Refinement:** The **(TSF) IT Environment** shall provide (**a**) **an encrypted** communication channel between itself and **(a remote trusted IT product) the TSF** that is logically distinct from other communication channels and provides assured identification of its end points and **(protection of the channel data from modification or disclosure) detection of the modification of data.**

FTP_TRP.1.1(1) - **Refinement:** The **(TSF) IT Environment** shall provide (**a**) an **encrypted** communication path between itself and ([**selection: remote, local**]) **the TSF** (**users**) that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from modification or disclosure.

FTP_TRP.1.2(1) - The **(TSF) IT Environment** shall permit *remote **users of the TSF*** to initiate communication **to the TSF** via the trusted path.

FTP_TRP.1.3(1) – **Refinement:** The **(TSF) IT Environment** shall **(require) initiate** the use of the trusted path for ***(initial)*** *user authentication*, ***all remote administration actions***, *[selection: [assignment: other services for which trusted path is required], none]*.

FTP_TRP.1.1(2) - **Refinement:** The **(TSF) IT Environment** shall provide (**a**) an **encrypted** communication path between itself and ([**selection: remote, local**]) (**users) the TSF** that is logically distinct from other communication paths and provides assured identification of its end points **(protection of the communicated data from modification or disclosure) and detection of the modification of data.**

FTP_TRP.1.2(2) - The **(TSF) IT Environment** shall permit *remote users **of the TSF*** to initiate communication **to the TSF** via the trusted path.

FTP_TRP.1.3(2) – **Refinement:** The **(TSF) IT Environment** shall **(require) initiate** the use of the trusted path for ***(initial)*** *user authentication*, ***all remote administration actions***, *[selection: [assignment: other services for which trusted path is required], none]*.