U.S. Government Biometric Verification Mode Protection Profile for Medium Robustness Environments

Information
Assurance
Directorate

Version 1.0

November 15, 2003

Protection Profile Title:

U.S. Government Biometric Verification Mode 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

1.0 INTRODUCTION

This Biometric Verification Mode Protection Profile (PP) for Medium Robustness Environments was sponsored by the Biometrics Management Office (BMO) and the National Security Agency (NSA). A verification mode biometrics device is one that *authenticates* a user for a claimed identity. This is distinctly different from an identification mode biometrics device, which attempts to identify an individual by their biometric characteristic. 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 Biometric Verification Mode Protection Profile (PP) for Medium Robustness Environments

Sponsor: The Biometrics Management Office and the 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 November 15, 2003

Keywords: Protection Profile, Medium Robustness Environments, verification mode, liveness, biometrics

1.2 Protection Profile Overview

This Protection Profile (PP) specifies the minimum functional and assurance security requirements for biometric products operating in verification mode to provide authentication allowing physical and logical access control to facilities as well as to information systems in medium robustness environments (see Section 3.0 for a characterization of medium robustness environments). Biometric systems are enabling technologies designed to augment existing security measures by positively authenticating individuals based on measurable physical features or behaviors. Due to the unique nature of a biometrics TOE and the desire of the PP authors to attempt to accommodate the wide range of biometric technologies, explicit requirements were necessary, as was a great deal of refinement of the CC requirements.

The requirements section of this PP specifies a need to protect biometric templates, to provide confidentially, and integrity. Since the biometric package (which includes the user identifier and their associated reference template(s)) may be stored in a device outside the control of the TOE, the biometrics TOE encrypts biometric packages for confidentiality reasons, and an enrolling TOE cryptographically signs a biometrics package so that modification of the package can be detected.

A TOE conformant to this PP satisfies the specified functional requirements, as well as the Medium Robustness assurance requirements that are expressed in Section 5.2 TOE Security Assurance Requirements. 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.

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 Related Protection Profiles

A basic robustness PP for a biometric TOE operating in verification mode has many of the same functional requirements, but does not require the use of cryptography to protect the biometric packages. Contrary to a medium robustness TOE, the basic robustness TOE has a reliance on the IT environment in order to address some of the threats and to enforce its security policies. The basic robustness PP has less stringent assurance requirements as well.

Rather than write a PP that specifies requirements for both verification mode and identification mode, a decision was made to write a PP for each mode of operation. This affords product developers the opportunity to evaluate their product and claim conformance to a PP if their product operates in only one of the modes of operation. This approach allows a product that operates in both modes the opportunity to claim conformance to each of the PPs. The following PPs make up the family of PPs sponsored by the BMO and NSA:[1]

- U.S. Government Biometric Verification Mode Protection Profile For Basic Robustness Environments, dated (TBD)
- U.S. Government Biometric Identification Mode Protection Profile For Medium Robustness Environments, dated (TBD)
- U.S. Government Biometric Identification Mode Protection Profile For Basic Robustness Environments, dated (TBD)

1.4 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 the word refinement in **bold text** and the added/changed words are in **bold text**. In cases where words from a CC requirement were deleted, a separate attachment indicates the words that were removed.

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).

[1] This is the first Protection Profile to be released in the family of Biometrics PPs and the remaining PPs are currently in draft form and not yet available for public release.

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.

1.5 Protection Profile Organization

Section 1, Protection Profile Introduction, provides 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, defines the security functional and assurance requirements derived from the Common Criteria, Part 2 and Part 3, respectively, that must be satisfied by the TOE and the Non-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

This section describes biometric authentication devices as the Target of Evaluation (TOE) for this protection profile.

Biometric TOEs are unlike other information-technology-related TOEs. Untrusted users who interact with the TOE (known as "subjects" in the biometrics community, but not in the Common Criteria community) are not really *users* of the TOE. Their only role is to present a claimed identity and a fresh biometric sample, and the biometric TOE decides whether the biometric sample comes from a live individual and whether the biometric sample matches the biometric previously enrolled by the user with the claimed identity. The TOE does not contain any user data and does not provide a logical interface to untrusted users. The TOE only contains TSF data and the logical interface presented is only for administrative functions.

The physical and logical boundaries of the TOE will differ depending upon a vendor's implementation and the intended use of the product. There are many permutations of where these components can be hosted.

For controlling physical access (e.g., a building or room), a TOE could be comprised of components that are physically and logically housed in a single unit. An example is a device whose ultimate purpose is to control access to a door, which performs the capture and comparison functions within a single unit and is stand alone. A TOE could also have multiple capture devices that transmit the live template to a server that then performs the comparison function, which then generates the match/no match decision.

For controlling local logical access to an IT product (e.g., a workstation) the TOE's physical boundary could take different forms as well. As with the example above, the TOE could be contained in a single unit and provide a match/no match decision to the IT product, or the TOE could be physically separated. If the TOE is physically separated it could use the IT product to transmit data (e.g., the live template, capture device's identity) through the IT product to another component of the TOE that performs the comparison function, which then in turn provides the match/no match decision to the IT product. It is important to note that the TOE includes all the hardware and software that play a role in the TOE being able to satisfy the security requirements specified in this PP. When the TOE is physically separated, cryptography is used to maintain confidentiality and to detect modification of the transmitted data. It is also important to note that none of the TOE's software is executing on a platform other than the trusted platform provided by the TOE. This means that the comparison software or any capture controller function is not running on an IT product other than the TOE. Figure 1 illustrates an example of a distributed TOE. In this example, the capture device is connected to an IT product (e.g., workstation) via a direct connection (e.g., USB connection) and the IT product is connected to a network. The capture device transmits the live template, and possibly other data (e.g., unique device id), to the comparator through a path that is not trusted with respect to the TOE. This is acceptable, since the capture device signs and encrypts the data being transmitted. The comparator retrieves the reference template from storage. The reference template is included in the biometric package, which is encrypted and cryptographically signed by the TOE (or another authorized entity). The comparator compares the templates and generates a match/no match decision, which is then sent to the IT product in the clear. Sending the decision in the clear is permitted, since once the decision leaves the TOE's scope of control, it is left to the IT environment, including the IT product, to handle the decision appropriately.

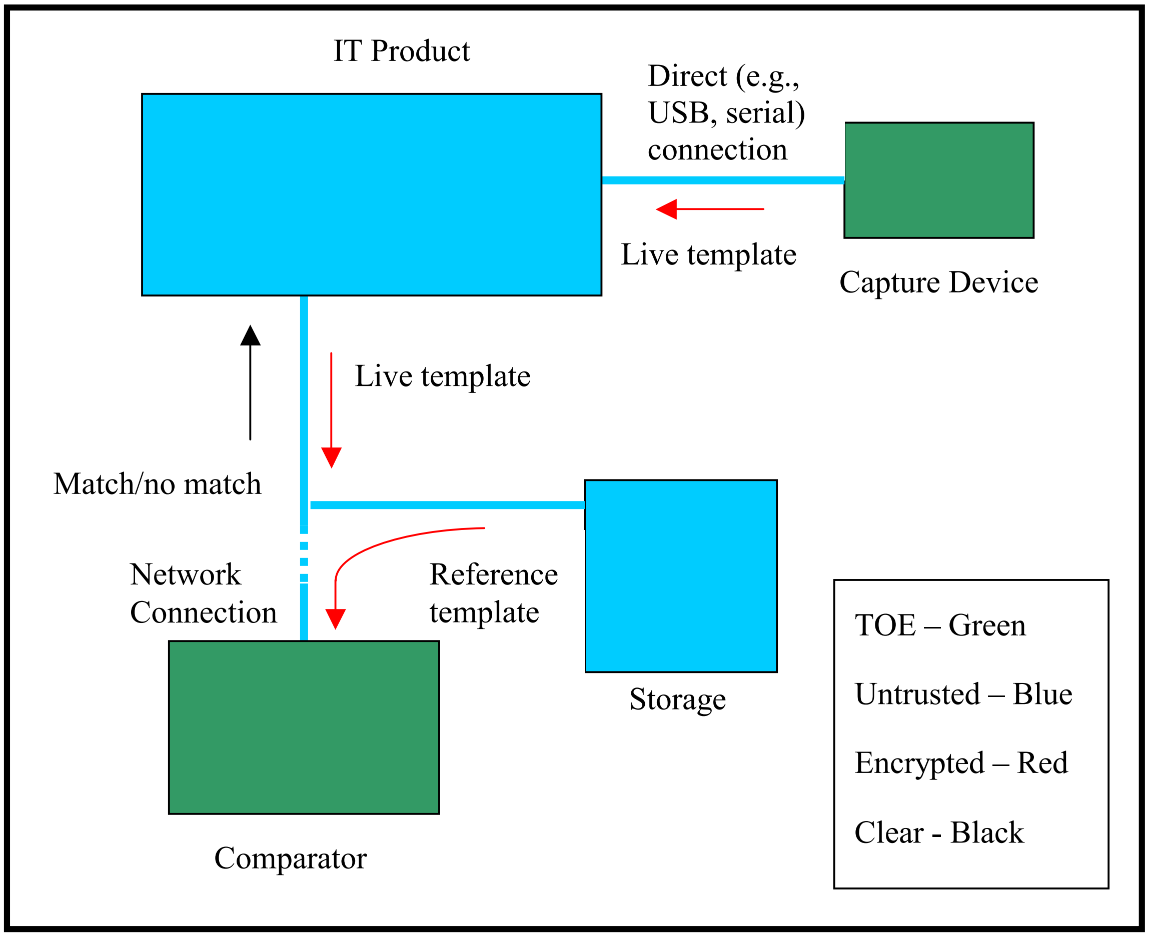


Figure 1. Example of a distributed TOE.

Another important aspect of the TOE, as defined by this PP, is that the storage of the biometric reference template is outside the scope of the TOE. This was done to allow flexibility in the deployment of the TOE and the scenarios under which the TOE or instantiations of the TOE may be used. The requirements in this PP were written to allow the storage of reference templates to take place at a single repository, be distributed across database servers, to allow single reference templates to be stored on a smart card, or other ways in which a developer wishes to handle storage. This is secure, since the biometrics package that contains the reference template is signed and encrypted by the TOE that performs the enrollment. However, the enrolling TOE must be a trusted signing authority for any instantiation of TOEs that are to use the biometric package when performing the authentication process.

This TOE requires that a second, non-biometric authentication mechanism (e.g., password, PIN) be available to end-users for administrative purposes. This was done to provide end-users with the flexibility of requiring more rigorous authentication for an administrator if they choose, or to allow administrators to solely use the non-biometric authentication mechanism. The latter may be useful if the capture device became unusable.

2.1 Biometric TOE Functionality

"Biometric Authentication" refers to the automatic identification or identity verification of living individuals based on physiological or behavioral characteristics. Examples of physiological characteristics include hand or finger images, facial characteristics, speaker verification and eye patterns. Biometric authentication is the "automatic", "real-time", "non-forensic" subset of the broader field of human identification.

In this protection profile, biometric devices are seen as components of security systems that provide positive authentication. As with other types of authentication technologies, biometrics provides mechanisms to quickly and securely associate an identity with a person. The distinctive feature about biometric technologies as an authentication factor is that the presenter of a valid biometric that matches an enrolled biometric is, by definition, an authorized user, in contrast with technologies such as tokens or passwords, where valid instances of these items can be presented by unauthorized users.

Figure 2 shows a simple model of a biometric TOE showing major components required for this protection profile. This figure, as well as Figure 3 and Figure 4, shows a one-way information flow that does not take into account the information flow from the TOE to the user (e.g., prompt for entering a claimed user identifier, or other information, such as directing the user which finger to present to the capture device). The following is a description of each block in the diagram:

- *Liveness Check & Capture* – A liveness check that determines if the host of the biometric sample has certain characteristics belonging to living human beings. In capture, a sample of the user's biometric is acquired using the required sensor (camera, microphone, fingerprint scanner, etc.). It is important to note the liveness check is performed at the same time as the capturing of the biometric characteristic.

- *Extraction* – Process by which the biometric sample captured in the previous block is transformed into an electronic representation. During enrollment this electronic representation is known as the biometric template. During the authentication process, it is known as the live sample.

- *Package Creation* – Performed only during enrollment. The TOE creates a "Biometrics Package" during enrollment. The biometric package is where a user identifier is associated with one or more reference templates. Cryptographically bind the user's identity and additional information with the biometric template to create a biometric package for storage.

- *Package Assurance* – Performed only during enrollment. Uses cryptographic methods to protect the confidentiality and integrity of the biometric package for storage.

- *Package Validation* – Performed only during authentication. Verifies the integrity of the biometric package received from storage and the validity of the signing authority.

- *Comparison* – Performed only during authentication. Matches the live sample and biometric templates. The result from the matching is a score, which is then compared against predefined threshold values.

- *Security Management Functions* – The TOE provides management functions to the TOE administrator that includes setting of the threshold, determining audit events, reviewing audit information, and key management.

This protection profile requires that when the matching score is outside the maximum and minimum threshold range, a *no-match* result is generated.

Cryptographic methods and modules must comply with approved standards and be validated by NIST's FIPS 140-2 validation program.

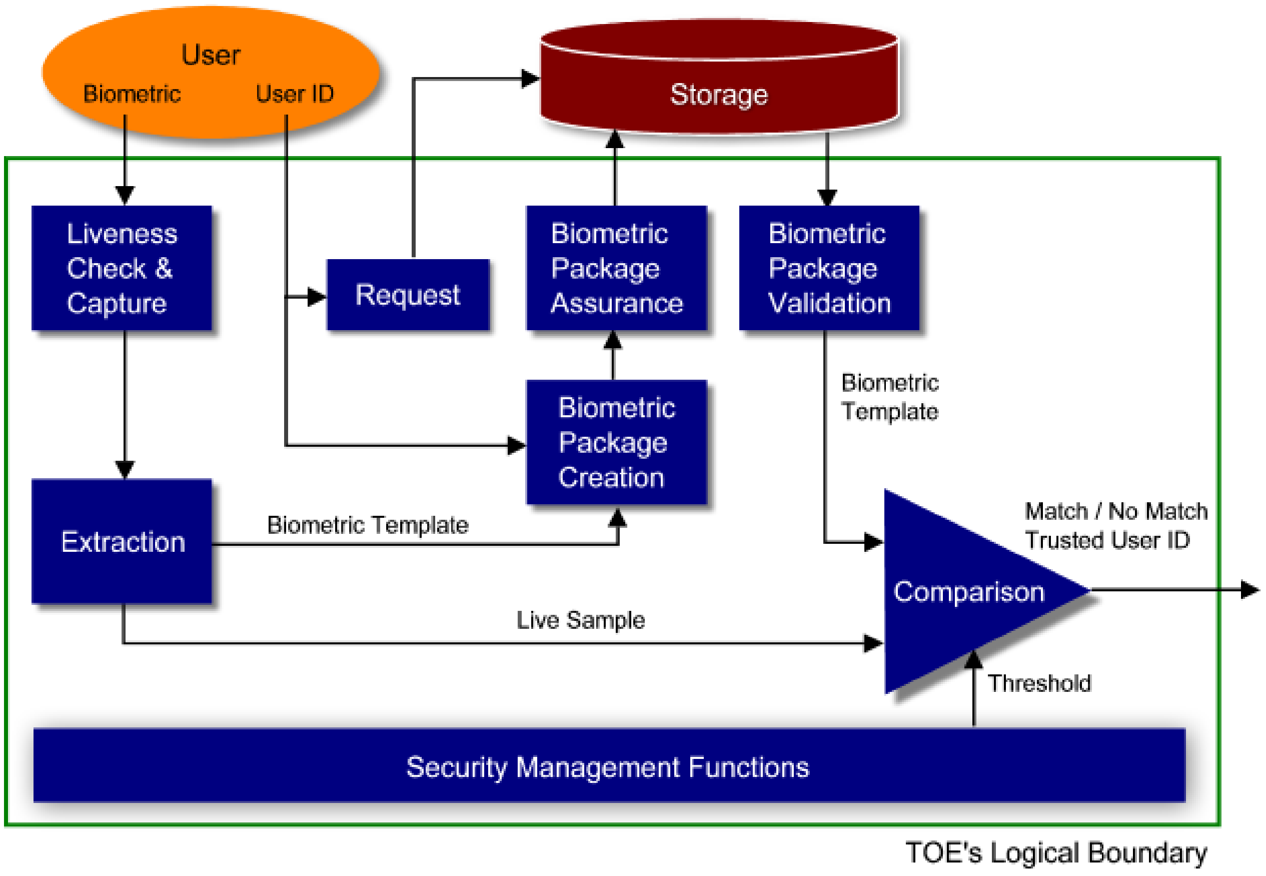


Figure 2. TOE functional block diagram

The basic processes a biometric TOE supports are enrollment and authentication. During enrollment, the biometric TOE captures the biometric sample from an enrollee, transforms it into a biometric template, and associates this template with the enrollee's identity for storage.

During authentication, the biometric TOE can be used for identification or verification of the person's identity. In identification, the biometric TOE attempts to determine the identity of a person by comparing the captured biometric sample against a database of enrolled templates for a match. In verification, the biometric device verifies a person's claimed identity by matching a captured biometric sample against the enrolled template associated with the claimed identity. This document considers a biometric TOE operating only in the verification mode.

The next sections describe the enrollment and verification modes in more detail.

2.1.1 The Enrollment Process

Figure 3 highlights the components of a biometric TOE involved during enrollment. Certainly, the process to enroll a user in the biometric TOE will form a part of a larger registration step.

The site should follow appropriate procedures for validating the identity of the individuals before enrolling them into their system. Only enrollment administrators can enroll users in a biometric TOE. The TOE's administrative guidance provides enrollment administrators guidance about acceptable quality metrics in regards to the quality of the biometric template.

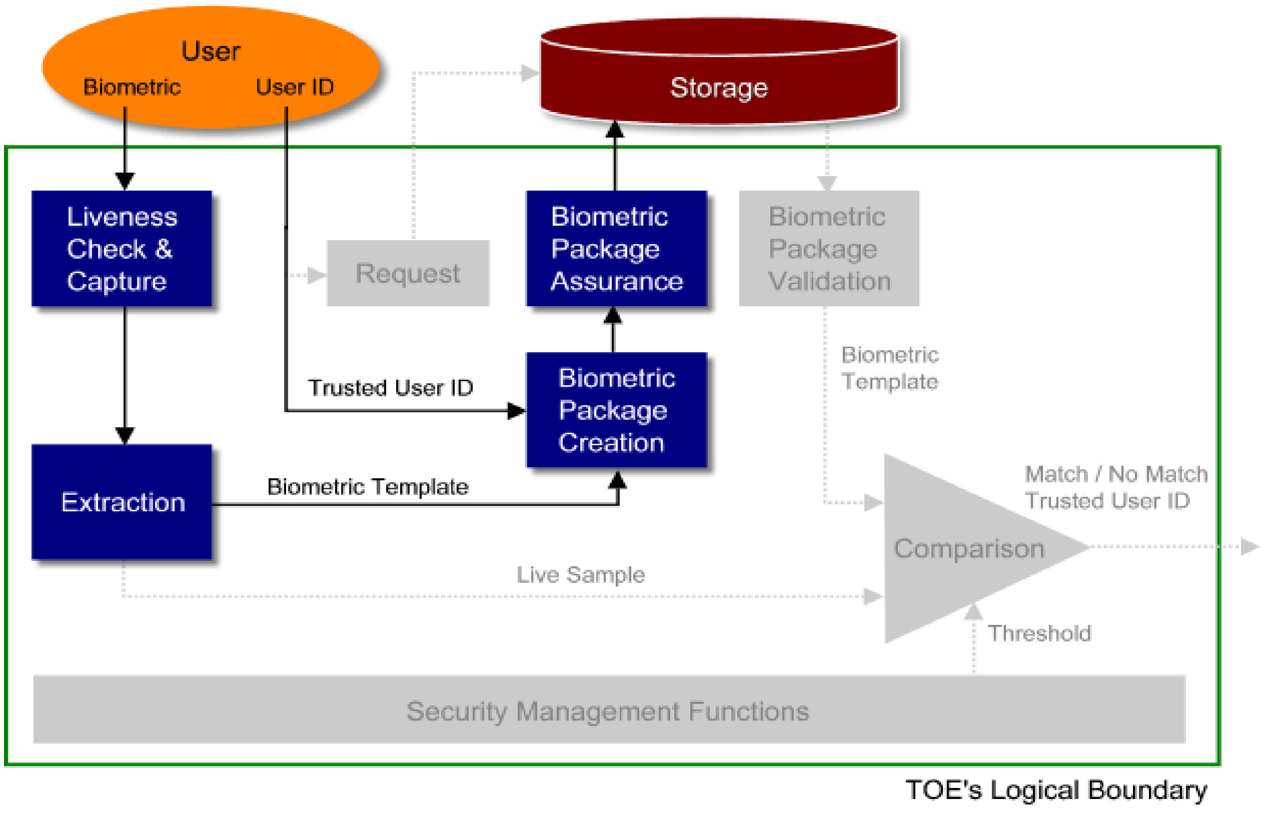


Figure 3. Block diagram of the enrollment process.

During enrollment, a biometric package is created that binds the *trusted user identifier* with the biometric template(s). It may include additional information if the TOE developer wishes, such as access privileges. After enrollment, the biometric package may be stored locally within the TOE, or on a storage device outside the TOE. The storage of biometric packages is outside the scope of this protection profile. Since the storage of the biometric packages is outside of the TOE's scope of control, cryptographic methods are used to ensure confidentiality of the biometric package is maintained, and to detect modification of the package. These methods are employed during the enrollment and verification processes.

2.1.2 The Verification Process

Figure 4 highlights the components of a TOE involved during verification. The TOE retrieves the biometric package of the user's *claimed identity* from storage, decrypts the package, and confirms that the TOE or a trusted signing authority has cryptographically signed the biometrics package.

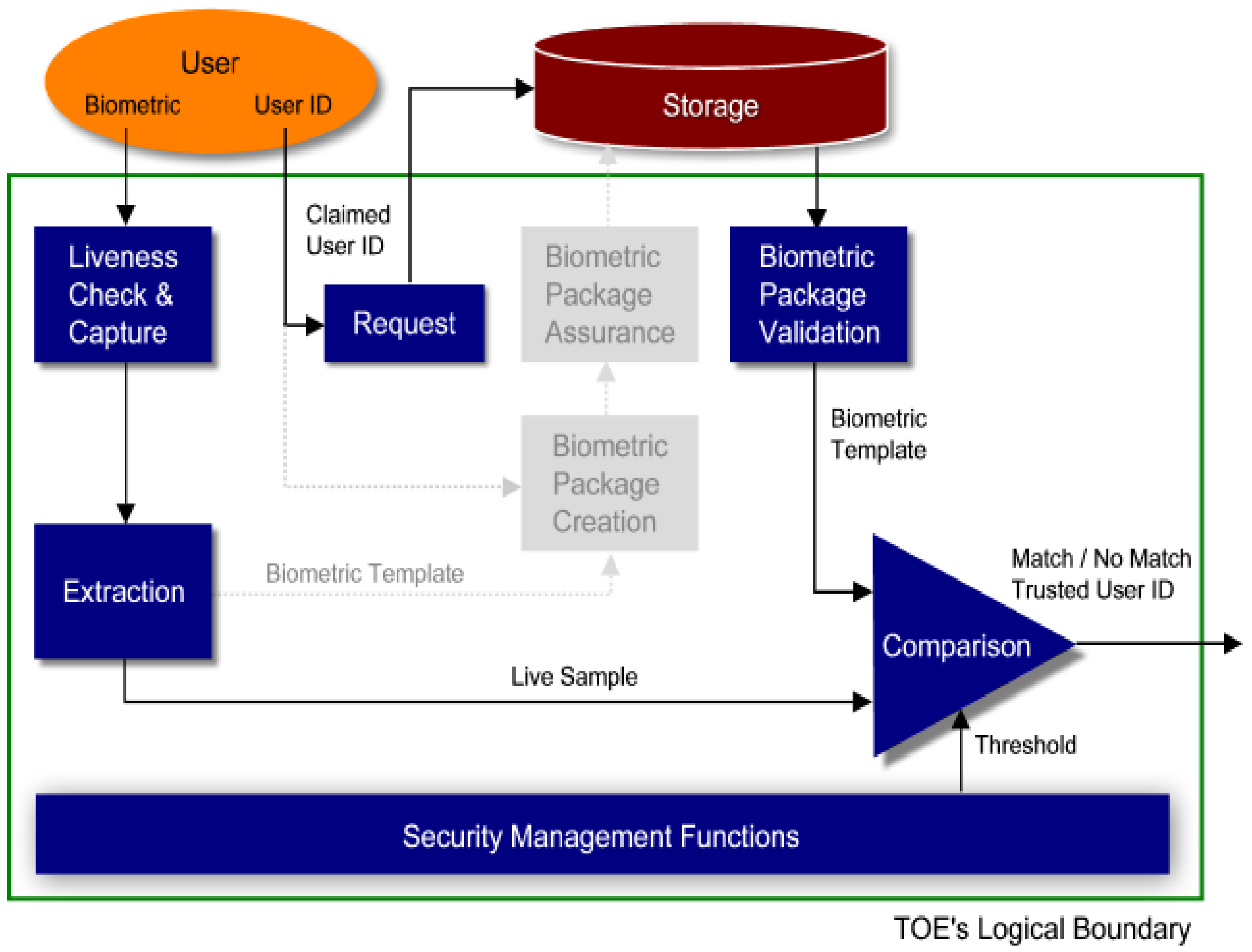


Figure 4. Verification process.

The biometric template(s) in the validated biometric package is then matched against a live sample captured from the user and a match/no-match result is generated. The Security Administrator can set a threshold range that determines the match/no-match result. However, the false acceptance and false rejection rates stated in this protection profile limit the range of acceptable values for the thresholds. The match/no-match result from the verification process is then passed to the IT environment, which will use the decision accordingly.

It is important to note the distinction between the *claimed user identifier* and *trusted user identifier*. The claimed user identifier is what the user presents to the biometrics TOE and is used to determine which biometric package to use in the verification process. The trusted user identifier is the identifier that is bound with the reference template in the biometrics package. This is a trusted user identifier, since the identity has been authenticated, whereas the claimed user identifier has not been authenticated. These two identifiers could be the same identifier (e.g., joe_user), but it is not required.

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 using 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 biometric TOE does not contain any user data that requires protection, but it may provide access to an entity with high value data. If the biometric device 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 (e.g., users, administrators, other IT systems) have with respect to the TOE (and thus the resources of that TOE, including the TOE itself) is an abstract concept reflecting a combination of the trustworthiness of an entity and the access and privileges granted to that entity with respect to the resources of the TOE. For instance, entities that have total authorization to all data on the TOE are at one end of this spectrum; these entities may have privileges that allow them to read, write, and modify anything on the TOE, including all TSF data. Entities at the other end of the spectrum are those that are authorized to few or no TOE resources. For example, in the case of 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 is authorized to certain data on a TOE, yet the owner connects 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.

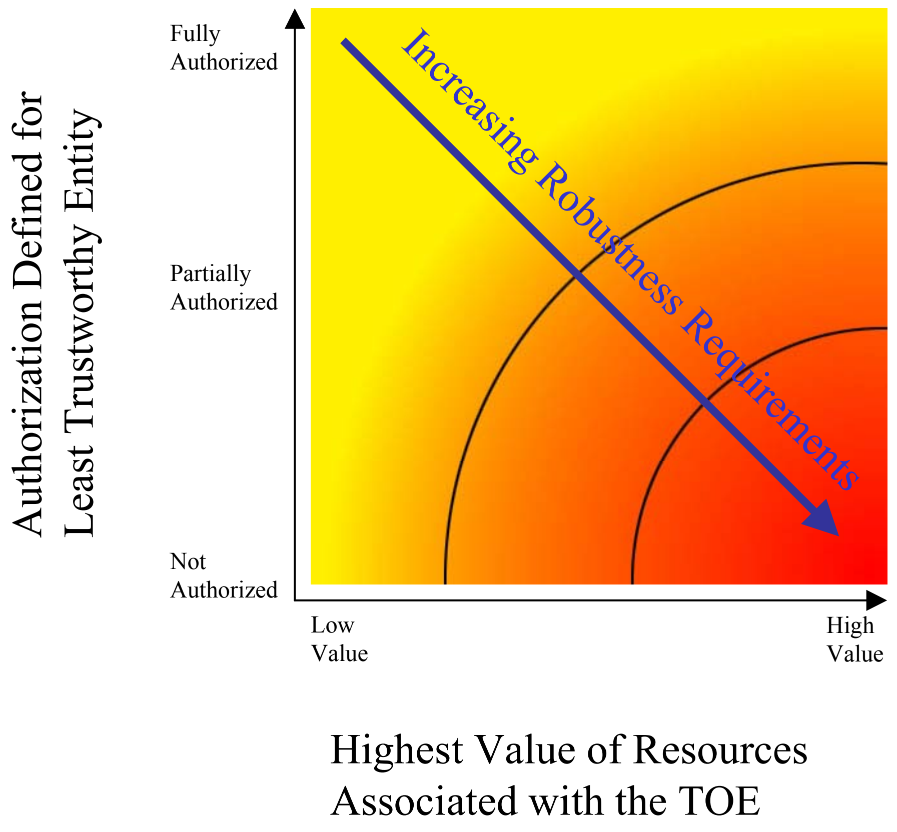
When assessing any environment with respect to Information Assurance the critical point to consider is the likelihood of an attempted security policy compromise, which was characterized in the previous section in terms of entity authorization and resource value. As previously mentioned, robustness is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

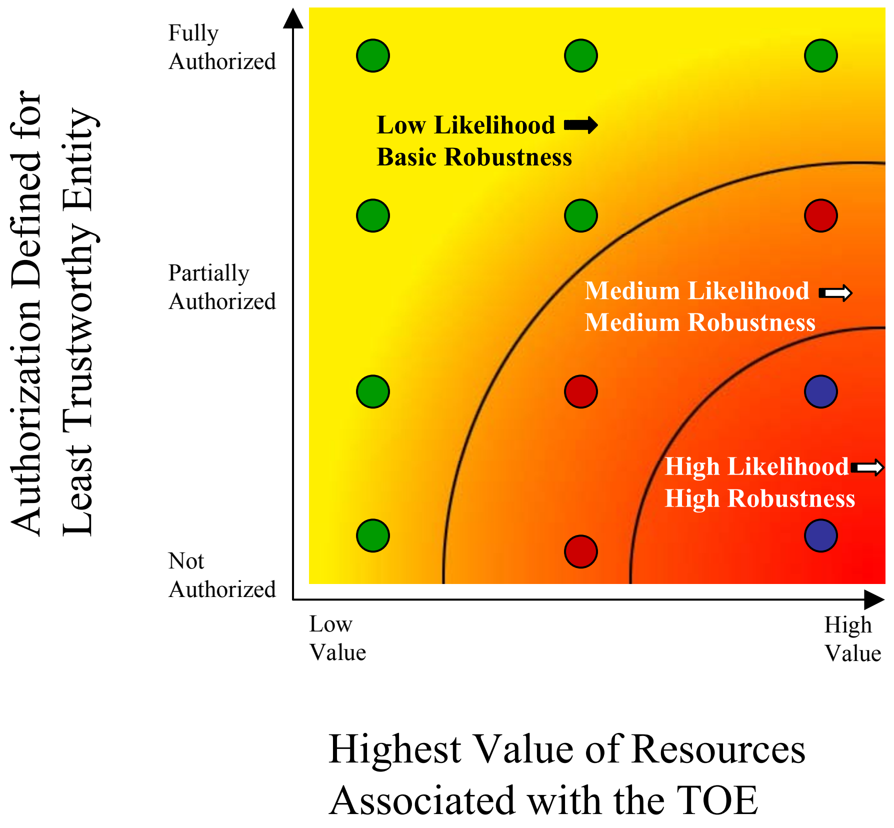
The second case is a TOE that processes high-value (e.g., classified) information. The organization requires that the TOE be stand-alone, and that every user with physical and logical access to the TOE undergo an investigation so that they are authorized to the highest value data on the TOE. Because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


Authorization Defined for Least Trustworthy Entity
Fully Authorized
Partially Authorized
Not Authorized
Increasing Robustness Requirements
Low Value
High Value
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 neither be practical nor particularly useful. Instead, in order to implement the robustness strategy where there are only three robustness levels: Basic, Medium, and High, the graph is divided into three sections, with each section corresponding to 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 3.6 of this PP, the targeted threat level for a medium robustness biometric device operating in a verification mode 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 biometric authentication device.

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:

1. Biometric specific environment issues;
2. Assumptions about the security aspects of a compliant TOE environment;
3. Threats to TOE which are addressed by the TOE; and
4. Organizational security policies that compliant TOEs must enforce.

3.4 Biometric TOE Environment

Biometric technology is somewhat different than other IT technologies in that the inputs to the TOE are not perfectly repeatable in practice. That is, one biometric sample from an individual will not be exactly the same as a corresponding sample from the same individual a few seconds or minutes (let alone years) later. Therefore certain performance requirements for the TOE are stated in terms of probabilities. These probabilities must account not only for variations in the TOE's performance, but also for natural variation in the inputs to the TOE.

The end-user must take into consideration the trade-offs between using a biometric device versus another form of authentication. Biometrics may offer a convenient means of authentication since users are not required to remember a password that is not easily guessable. Biometrics also offers an advantage in that it may be more difficult to perform a brute force attack against a user's account than with a password mechanism. The maximum false acceptance rate (1×10^{-6}) for this TOE is weaker than the probability that a password can be guessed (1×10^{-8} for the non-biometric authentication mechanism in this PP). But it may be much more difficult to prepare and present 10^6 different biometric samples than it is to enter 10^8 passwords.

However, the degree of assurance in the authentication of an individual using biometric technologies varies. In order to accommodate a wide range of technologies this PP mandates a maximum false acceptance rate. End-users should pay close attention to the provided selection in the FIA_SOS.2 requirement, as this requirement affords a product developer the ability to provide a lower false acceptance rate if appropriate for their product. Another varying factor in the quality of the authentication decision is ability of the TOE to perform a check for liveness. Various technologies may be limited in their ability to perform a liveness check and the end-user should consider this when determining the suitability of a biometric product. The FIA_UAU.5 requirement should be considered when comparing products, as the product developer fills in an assignment that states how their product performs a check for liveness of the biometric characteristic being presented to the capture device. The PP authors could not specify what takes place during a liveness test, due to the varying biometric technologies and the state of technology increasingly improving in this area.

3.5 Assumptions

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

A.ENROLLMENT_APPROVAL	It is assumed that sites follow appropriate procedures for validating the identity of enrolled individuals.
A.NO_GENERAL_PURPOSE	There are no general-purpose computing or storage repository capabilities (e.g., compilers, editors, or user applications) available on the TOE.
A.OPERATING_RANGE	The TOE is placed in an environment that does not exceed its normal operating range (e.g., temperature, humidity) as defined by the vendor.

3.6 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:

1. The motivation for the threat agent defines the upper bound with respect to the level of robustness required for the TOE**.**

2. 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**).**

3. 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.

It is important to note that while some of the threats listed in this PP are the same as though listed in the Biometric Verification Mode PP for Basic Robustness they are not necessarily countered or mitigated in the same manner or to the same degree. The rationale section of the PP provides the details of how a threat is countered/mitigated.

3.6.1 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, emissions occurring during enrollment that would allow an eavesdropper to reconstruct either the biometric sample or the generated template) and it is up to a site to determine how these types of threats apply to its environment.

T.ADMIN_ERROR	An administrator may incorrectly install or configure the 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.BYPASS	An attacker may bypass any component of the biometric product and gain unauthorized authentication.
T.CRYPT_ATTACK	An attacker may defeat security functions through a cryptographic attack against the algorithm, through cryptanalysis on encrypted data, or through a brute-force attack and thereby gaining unauthorized authentication.
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.HIGH_QUALITY_ARTIFACT	An attacker may use a high quality artifact (e.g., artificial hand/fingerprint, life-size photograph, or other synthetic means) to gain unauthorized authentication.
T.MIMIC	An attacker may masquerade as an enrolled user by presenting their biometric characteristic that is similar, or by reproducing the biometric characteristics of the enrolled user (e.g., changing his/her voice, forging a signature, or other mean of mimicry) to gain unauthorized authentication.
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.CORRUPTED_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 thereby causing potential security vulnerabilities.
T.REPLAY_RESIDUAL_IMAGE	An attacker may attempt to "reuse" an authorized user's biometric residual characteristic (e.g., finger print left on capture device) to gain unauthorized access.
T.RESIDUAL_DATA	Residual biometric authentication data from a previous valid user if not cleared from memory may allow an attacker to gain unauthorized authentication.
T. REFERENCE_TEMPLATE	An attacker modifies or creates a biometric reference template in storage or transmission to/from storage to gain unauthorized authentication.

T.POOR_ENROLLMENT	An attacker may direct an attack against a low quality reference template and gain unauthorized authentication.
T.TAMPER	An attacker may modify or otherwise alter the software or hardware components, the connections between them thereby gaining unauthorized authentication.
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	An attacker may gain unauthorized access to an administrator's unattended session.
T.UNAUTHORIZED_ACCESS	A user may gain access to administrative functions 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.

3.7 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.CRYPTOGRAPHIC_ FUNCTIONS	The TOE shall provide cryptographic functions (i.e., encryption/decryption and digital signature

	operations) to maintain the confidentiality and allow for detection of modification of TSF data that is transmitted between physically separated portions of the TOE, or stored outside the TOE.
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_MULTIPLE_ROLE	The TOE will provide multiple administrative roles to isolate non-overlapping administrative functions.
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.AUTHENTICATION	The TOE will provide a biometric authentication mechanism to authenticate users for the IT environment or non-IT environment.
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 at a customer's site.
O.CRYPTOGRAPHIC_ FUNCTIONS	The TOE shall provide cryptographic functions (i.e., encryption/decryption and digital signature

	operations) to maintain the confidentiality and allow for detection of modification of TSF data that is transmitted between physically separated portions of the TOE, or stored outside the TOE.
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.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.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.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 non-technical or procedural means. All of the assumptions stated in Section 3.4 are considered to be security objectives for the environment. The mapping and rationale for the security objectives are described in Section 6.

OE.ENROLLMENT_APPROVAL	Sites follow appropriate procedures for validating the identity of enrolled individuals.
OE.NO_GENERAL_PURPOSE	There are no general-purpose computing or storage repository capabilities (e.g., compilers, editors, or user applications) available on the TOE.
OE.OPERATING_RANGE	The TOE is placed in an environment that does not exceed its normal operating range (e.g., temperature, humidity) as defined by the vendor.

5.0 IT SECURITY REQUIREMENTS

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

5.1 TOE Functional Security Requirements

This section provides functional and assurance requirements that must be satisfied by a PP-compliant TOE. These requirements consist of components from the CC Part 2 and Part 3, NIAP interpreted requirements, and explicit requirements. Table 5.1 summarizes the TOE Functional Requirements to meet the stated objectives. Table 5.2 identifies the explicit requirements that were necessary to express the desired functionality.

As a vehicle for providing a further understanding of and context for security requirements, *Application Notes* have been selectively added to this PP. When they appear in the text, these follow either an element, a component, or set of components. In certain cases, the SFRs need interpretation to deal with particular characteristics of biometric systems or to convey the PP author's intent of an SFR, including any left open assignments or selections. Advice on interpretation is provided in the form of application notes where the authors felt it appropriate.

Table 5.1 - Security Functional Requirements

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 symmetric keys using RNG)
FCS_CKM.4	Cryptographic Key Destruction
FDP_RIP.2	Full residual information protection
FIA_ATD.1	User attribute definition
FIA_SOS.1	Verification of secrets
FIA_SOS.2	TSF Generation of secrets

Functional Components (from CC Part 2)	
FIA_UAU.2	User authentication before any action
FIA_UAU.5	Multiple authentication mechanisms
FIA_UAU.7	Protected authentication feedback
FIA_UID.2	User identification before any action
FMT_MOF.1(1)	Management of security functions behavior (audit selection)
FMT_MOF.1(2)	Management of security functions behavior (audit review)
FMT_MOF.1(3)	Management of security functions behavior (alarms)
FMT_MOF.1(4)	Management of security functions behavior (TSF non-Cryptographic Self-test)
FMT_MOF.1(5)	Management of security functions behavior (Cryptographic Self-test)
FMT_MOF.1(6)	Management of security functions behavior (Maintenance Mode)
FMT_MOF.1(7)	Management of security functions behavior (Enrollment)
FMT_MOF.1(8)	Management of security functions behavior (non-biometric Authentication Mechanism)
FMT_MOF.1(9)	Management of security functions behavior (Biometric Authentication Mechanism)
FMT_MTD.1(1)	Management of TSF data (cryptographic TSF data)
FMT_MTD.1(2)	Management of TSF data (time TSF data)
FMT_MTD.1(3)	Management of TSF data (Authentication Mechanism Data)
FMT_REV.1	Revocation
FMT_SMR.2	Restrictions on security roles
FPT_ITT.1(1)	Basic internal TSF data transfer protection (from disclosure)
FPT_ITT.1(2)	Basic internal TSF data transfer protection (from undetected modification)

Functional Components (from CC Part 2)	
FPT_PHP.3	Resistance to physical attack
FPT_RVM.1	Non-bypassability of the TSP
FPT_SEP.2	SFP domain separation
FPT_STM.1	Reliable time stamps
FTA_SSL.3	TSF-initiated termination
FTA_TAB.1	Default TOE access banners
FTA_TSE.1	TOE session establishment

Table 5.2 - Explicit Security Functional Requirements

Explicit Functional Components	
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 using AES)
FCS_COP_EXP.3	Cryptographic Operation (Digital Signature Generation/Verification)
FCS_COP_EXP.5	Cryptographic Operation (Random Number Generation)
FCS_COP_EXP.6	Cryptographic Operation (Cryptographic Hashing Function)
FIA_ENROLL_EXP.1	Enrollment
FMT_MTD_EXP.1	Management of TSF data (Capture device unique identifier)

Explicit Functional Components	
FPT_ITC_EXP. 1	TSF confidentiality
FPT_ITI_EXP.1	TSF detection of modification
FPT_PHP_EXP.1	Detection of physical attack
FPT_TST_EXP.4	TSF testing (with cryptographic integrity verification)
FPT_TST_EXP.5	Cryptographic self-test
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_SEL.1-NIAP-0407	Selective audit
FAU_STG.1-NIAP-0423	Protected audit trail storage
FAU_STG.NIAP-0414-1-NIAP-0429	Site-Configurable Prevention of Audit Loss
FIA_AFL.1-NIAP-0425(1)	Authentication failure handling (Against a single non-administrative user identifier)
FIA_AFL.1-NIAP-0425(2)	Authentication failure handling (Consecutive failed attempts)
FIA_AFL.1-NIAP-0425(3)	Authentication failure handling (Administrator Users)
FIA_USB.1-NIAP-0415	User-subject binding
FPT_RCV.2-NIAP-406	Recovery from Failure

5.1.1 Security Audit (FAU)

FAU_ARP.1 Security alarms

FAU_ARP.1.1 – **Refinement:** The TSF shall

a) [generate an alarm by [assignment: method determined by the ST Author to generate the alarm],

b) block any further authentication attempts until the Security Administrator defined time period has elapsed, or an action is taken by the Security Administrator,

c) stop ongoing and prevent further enrollment activity until the Security Administrator takes some action,]

upon detection of a potential security violation.

Application Note: The TOE generates an alarm by a method determined by the ST Author. Acceptable methods may include sending an email, paging the Security Administrator, sending a message to an administrative console, sounding an audible alarm (e.g., bell, siren) or providing a visual alarm, such as a flashing light. The intent of this requirement is to alert an administrator that the TOE has encountered a potential security violation. While some implementations may provide an alarm that communicates an alarm condition more effectively to an administrator than other implementations, the PP does not want to exclude devices that may not be able to "immediately alert" an administrator (e.g., stand alone TOEs with no connectivity). The intent in b) is to provide the Security Administrator the choice of preventing the TOE from authenticating users until the Security Administrator takes some action (e.g., enable the TOE to perform authentication, clear the alarm and the TOE implicitly can resume performing authentication), or define a time period in which the TOE can begin performing authentication again. The time period should allow the flexibility of allowing the administrator to "throttle" throughput (e.g., a few minutes) or to assess the alarm and take the appropriate action (e.g., a few hours). The TOE may additionally send an alarm to the host IT environment to signify a potential security violation, but simply signaling the IT environment does not satisfy the intent of this requirement.

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 5.3**; 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].

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 chose "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..

Table 5.3 Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FAU_ARP.1	Potential security violation was detected	Identification of the event(s) caused the generation of the alarm
FAU_GEN.1-NIAP-0410	None	
FAU_GEN.2-NIAP-0410	None	
FAU_SAA.1-NIAP-0407	Attempts to enable/disable of any of the analysis mechanisms	The trusted user identity of the administrator performing the function
FAU_SAR.1	Attempts to open the audit trail	The trusted user identity of the administrator performing the function
FAU_SAR.2	Attempts to read information from the audit records	The trusted user identity of the administrator performing the function
FAU_SAR.3	None	
FAU_SEL.1-NIAP-0407	Attempts to modify the audit configuration	The trusted user identity of the administrator performing the

Requirement	Auditable Events	Additional Audit Record Contents
		function
FAU_STG.1-NIAP-0423	Attempts to backup and delete the audit trail	The trusted user identity of the administrator performing the function
FAU_STG.3	Reaching the defined percentage of storage capacity; Actions taken due to exceeding the threshold	The Audit Administrator defined percentage of storage capacity; The action to be taken if the audit trail becomes full
FAU_STG.NIAP-0414-1-NIAP-0429	None	
FCS_BCM_EXP.1	None	
FCS_CKM.1	Failure of the activity	
FCS_CKM_SYM_EXP.1	Failure of the activity	
FCS_CKM_ASYM_EXP.1	Failure of the activity	
FCS_CKM.4	None	
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	Type of cryptographic operation

Requirement	Auditable Events	Additional Audit Record Contents
	operation	Any applicable cryptographic mode(s) of operation, excluding any sensitive information
FDP_RIP.2	None	
FIA_AFL.1-NIAP-0425(1)	Reaching the specified number of failed authentication attempts; The action (e.g. disabling of an account, timeout) taken; The subsequent, if appropriate, restoration to the normal state (e.g. re-enabling of an account)	Claimed identity of the unsuccessfully authenticated user; Trusted user identity of the Security Administrator (if applicable) that took action to re-enable an account; Period of timeout (if applicable)
FIA_AFL.1-NIAP-0425(2)	Reaching the specified number of failed authentication attempts; The action (i.e., disabling of authentication at the offending capture device, timeout) taken; The subsequent, if appropriate, restoration to the normal state (e.g. re-enabling of authentication at the capture device)	Claimed identity of the unsuccessfully authenticated user(s)[2]; Trusted user identity of the Security Administrator (if applicable) that took action to re-enable an account; Period of timeout (if applicable)
FIA_AFL.1-NIAP-0425(3)	Reaching the specified number of failed authentication attempts; The action (e.g. disabling of an account, timeout) taken; The subsequent, if appropriate,	Claimed identity of the unsuccessfully authenticated administrator; Trusted user identity of the Security Administrator (if applicable) that took action to re-enable an account;

[2] For this requirement, there may be multiple user identifiers associated with this event, and the audit record contains all user identifiers that generated the event.

Requirement	Auditable Events	Additional Audit Record Contents
	restoration to the normal state (e.g. re-enabling of an account)	enable an account; Period of timeout (if applicable)
FIA_ATD.1	None	
FIA_UAU.1	None	
FIA_SOS.1	None.	
FIA_SOS.2	None.	
FIA_UAU.2	None.	
FIA_UAU.5	All use of the authentication mechanism(s)	Claimed identity of the user attempting to authenticate using the biometric authentication mechanism; Trusted user identifier of a successfully authenticated user; Unique identity of the capture device[3]; Result of liveness check; Identity of the IT entity that digitally signed the biometrics package; Comparison score of a non-match decision; Identity of the user presented at the non-biometric authentication mechanism

[3] The TOE has the ability to uniquely identify the capture device. If the TOE has multiple capture devices this identifier aids the Administrator in determining where the offending action took place. The unique identifier could be a identifier that is transmitted to the TOE, or could be associated with the capture device based on how it is connected to the TOE (e.g., a capture device is uniquely assigned to a physical port on a server component of the TOE).

Requirement	Auditable Events	Additional Audit Record Contents
FIA_UAU.7	None.	
FIA_UID.2	All use of the user identification mechanism, including the user identity provided	
FIA_USB.1-NIAP-0415	Success and failure of binding of user security attributes to a subject	The trusted user identity of the user whose attributes are attempting to be bound
FMT_MOF.1(1)	All attempts to enable, disable, determine, or modify the behavior of the audit generation functions in the TSF	The trusted user identity of the administrator performing the function
FMT_MOF.1(2)	All attempts to enable, or modify the behavior of the audit review functions in the TSF	The trusted user identity of the administrator performing the function
FMT_MOF.1(3)	All attempts to modify the behavior of the alarm and analysis functions in the TSF	The trusted user identity of the administrator performing the function
FMT_MOF.1(4)	All attempts to modify the behavior of the self-tests functions in the TSF	The trusted user identity of the administrator performing the function
FMT_MOF.1(5)	All attempts to enable or disable the cryptographic self-tests after key generation in the TSF	The trusted user identity of the administrator performing the function
FMT_MOF.1(6)	None	
FMT_MOF.1(7)	All attempts to determine, or modify the behavior of the enrollment functions in the TSF	The trusted user identity of the administrator performing the function
FMT_MOF.1(8)	All attempts to enable and disable the non-biometric authentication mechanism	The trusted user identity of the administrator performing the function
FMT_MOF.1(9)	All attempts to modify or determine the behavior of the biometric authentication	The trusted user identity of the administrator performing the function;

Requirement	Auditable Events	Additional Audit Record Contents
	mechanism	function; Any state change (enable/disable) of the liveness check
FMT_MTD.1(1)	All attempts to modify the cryptographic security data	The trusted user identity of the administrator performing the function
FMT_MTD.1(2)	All attempts to set the time and date used to form the time stamps	The trusted user identity of the administrator performing the function
FMT_MTD.1(3)	All attempts to query and set the authentication mechanism data	The trusted user identity of the administrator performing the function
FMT_MTD_EXP.1	All attempts to set the capture device identifier, if applicable[4]	The trusted user 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 trusted user identity of the administrator performing the function
FMT_SMR.2	All attempts to modify the group of users that are associated with a role	Trusted user identifiers that are associated with the modifications The trusted user identity of the administrator performing the function
FPT_ITT.1(1)	None	▨
FPT_ITT.1(2)	None	▨

[4] If the TOE does not provide the capability to set the capture device identifier (i.e., the TOE hardwires a unique identifier in the capture device) then this audit event is not applicable.

Requirement	Auditable Events	Additional Audit Record Contents
FPT_PHP.3	None	
FPT_RCV.2-NIAP-406	The fact that a failure or service discontinuity occurred; Resumption of the regular operation;	Type of failure or service discontinuity
FPT_RVM.1	None	
FPT_SEP.2	None	
FPT_STM.1	Changes to the time and date	Previous time and date; New time and date
FTA_SSL.3	The termination of a remote session by the session locking mechanism	The trusted user identity of the administrator associated with the session that was terminated
FTA_TAB.1	None	
FTA_TSE.1	All attempts at establishment of an administrator session	The claimed identity of the user attempting to establish the session For unsuccessful attempts, the reason for denial of the establishment attempt
FIA_ENROLL_EXP.1	All attempts to create a reference template, refreshing reference templates, or adding additional reference templates to a biometric package; All attempts to modify a reference template while resident in the TOE;	Trusted user identity of the administrator attempting to create/modify a reference template; The enrolled user's user identifier.
FPT_ITC_EXP. 1	Any failure to decrypt a biometric package	Claimed user identifier of the associated biometric package
FPT_ITI_EXP.1	Detection of modification of the biometric package	User identifier of the associated biometric package

Requirement	Auditable Events	Additional Audit Record Contents
FPT_PHP_EXP.1	Exposure of internal TOE components.	
FPT_TST_EXP.4	TSF testing (with cryptographic integrity verification)	Self-test that failed; The affected TSF components, including the TSF software and TSF data where modification was detected
FPT_TST_EXP.5	Cryptographic self-test	Self-test that failed;

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 **(if applicable)**; 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 in Table 5.3*.

Application Note: A subject identity is distinct from a user identifier (trusted or claimed). A subject identity is typically an active entity that is acting on behalf of a user (e.g., a process, in which case the process id would be the subject identity). In general, this subject may be a trusted subject or an untrusted subject. In this TOE there are two types of users: the untrusted users, which only have limited access to the TOE (i.e., present their biometric characteristic to the capture device); and trusted users, which are the administrators that administer the TOE. Since the untrusted users have limited interaction with the TOE, this TOE only has trusted subjects. The intent of requiring the identity of a trusted subject resulting from an authentication event is to provide information on which authentication mechanism(s) was used. The thought is that the biometric authentication mechanism(s) and the additional administrator authentication mechanism may have distinct subject identities, which could provide the Audit Administrator valuable information. In limited cases the subject identity or outcome does not apply (i.e., FPT_PHP_EXP.1)

FAU_GEN.2-NIAP-0410 User Identity Association

FAU_GEN.2.1-NIAP-0410 –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.

Application Note: The claimed user identifier may not be associated with a biometrics package (e.g., an invalid claimed user identifier was presented), however, the supplied claimed user identifier is captured in the audit record. This requirement applies somewhat differently depending on the type of user (i.e., untrusted user, administrator). For untrusted users, the TOE associates auditable events to a claimed user identifier that is supplied when a user attempts to authenticate. This claimed identifier may not be the same as the trusted user identifier (the one bound to the reference template) and this case is different than administrative users, because the TOE may have no knowledge of the human user associated with the supplied user identifier. This is because untrusted users may have been enrolled on a different TOE. However, the TOE is always able to associate the trusted user identifier of administrators with human users, since administrative users are "registered" in the TOE as required by FIA_ATD.1.

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) Accumulation of [

- a Security Administrator specified number of authentication failures against a single non-administrative user identifier,
- a Security Administrator specified number of consecutive failed authentication attempts,
- a Security Administrator specified number of authentication failures against an administrative user identifier;

b) Any failure of the cryptographic self-tests;

c) Any failure of the other TSF self-tests;

d) Any failure to generate a cryptographic key;

e) Detection of physical attack;

f) Any failure to decrypt a biometrics package;

g) Detection of modification of a biometrics package];

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

Application Note: The intent of this requirement is that an alarm is generated (FAU_ARP.1) once the threshold for the event in (a) 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 the iterations of FIA_AFL.1.1-NIAP-0425(1) – (3).

FAU_SAR.1 Audit review

FAU_SAR.1.1 - The TSF shall provide [the Audit Administrator and Security Administrator] with the capability to read [all audit information] from the audit records.

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

FAU_SAR.2 Restricted audit review

FAU_SAR.2.1 – **Refinement**: The TSF shall prohibit all users read access to the audit records, except **the Audit Administrator and Security Administrator**.

FAU_SAR.3 Selectable audit review

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

a) [user identifier;

b) subject identity;

c) reference template creation;

d) ranges of one or more: dates, times;

e) events that generate an alarm; and

f) [selection: [assignment: other criteria determined by the ST Author], no additional criteria]].

Application Note: The Audit Administrator and Security Administrator are the only users who can perform these functions, since they are the only users with read access to the audit records in the audit trail. Audit data should be capable of being searched and sorted on all criteria specified in a – e, 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 time period presented together to facilitate their audit review. In item (e), these are the events specified in FAU_SAA.1. 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 Audit 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) [success of auditable events;

d) failure of auditable events; and

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

Application Note: "event type" is to be defined by the ST author; the intent is to be able to include or exclude classes of audit events. While the Audit Administrator has the capability to "pre-select" audit events, this does not mean that the Audit Administrator has the capability to implicitly disable alarm events (FAU_SAA.1). If the Audit Administrator de-selects an event listed in FAU_SAA.1 that event will still generate an alarm if the Security Administrator has enabled that event(s) to generate an alarm.

FAU_STG.1-NIAP-0423 Protected audit trail storage

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

FAU_STG.1.2-NIAP-0423 - **Refinement**: The TSF shall *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 [generate an alarm by [assignment: method determined by the ST Author to generate the alarm]], if the audit trail exceeds [an Audit Administrator settable percentage of storage capacity].

Application Note: As with FAU_ARP.1, the TSF generates an alarm to indicate that the audit trail has reached the Audit Administrator defined percentage of storage capacity.

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 **Audit Administrator** the capability to select one of the following actions: *prevent auditable events, except those taken by the Audit Administrator, overwrite the oldest stored audit records* **or** [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.

Application Note: The TOE provides the Audit Administrator the option of preventing audit data loss by preventing auditable events from occurring, except those actions taken by the Audit Administrator. This means that **only** the Audit Administrator can successfully be authenticated. The Audit Administrator actions under these circumstances are not required to be audited, since a user acting in this role will have to perform some action to manage the audit trail and address the problem. The TOE also provides the Audit Administrator the option of overwriting "old" audit records rather than preventing auditable events, which may protect against a denial-of-service attack.

FAU_STG.NIAP-0414-1.2-NIAP-0429 **Refinement**: The TSF shall **as a default** *prevent auditable events, except those taken by the* ***Audit Administrator*** if the audit trail is full.

Application Note: While FAU_STG.NIAP-0414-1.1-NIAP-0429 provides the Audit Administrator with the capability to select the TOE's behavior when the audit trail is full, this requirement ensures that as a default, audit records are not lost when the audit trail becomes full.

5.1.2 Cryptographic Support Requirements (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 140-2 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.[5] 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 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* is used 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 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 cryptographic requirements are structured to accommodate use of FIPS 140-2-validated cryptomodules in meeting the requirements. Since the FIPS 140-2 scheme does not cover all aspects of all algorithms, a convention is needed to distinguish the cryptographic functionality that the TSF is required to provide that cannot be provided by a FIPS-validated cryptographic module (e.g., Diffie-Hellman Key Agreement) from cryptographic functionality that can be provided via a FIPS-validated cryptomodule (e.g., AES). In the following text and requirements, "cryptomodule" is used in the very specific sense that it is:

[5] 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.

- a module that is FIPS 140-2 validated (to comply with FCS_BCM_EXP below);
- 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 for the cryptomodule.

Further, when the requirements mandate that a FIPS-approved security function be used, it requires that that security function is implemented in a cryptomodule as defined above.

It is the intent of these requirements (and the requirements are worded such) that whenever cryptographic functionality that can be FIPS-validated is required, that functionality be implemented in a cryptomodule. This means that when key management requirements (including key generation) are present, the key management functionality must be present in the cryptomodule. As an example, cryptomodules implementing AES must generate their own key.

FCS_COP_EXP.1(5) (Cryptographic Operation: Random Number Generation) is unusual because it is not a FIPS-approved security function as listed in Annex A to FIPS 140-2. However, its inclusion in the set of requirements mandates that whenever random number generation is required by a cryptographic function (e.g., generation of symmetric key, generation of the private key of a public-private key pair) that it be implemented in a cryptomodule.

FCS_COP_EXP.1(6) (Cryptographic Operation: Cryptographic Hashing Function) is similar because it is used by many other cryptographic operations (e.g., digital signature generation and verification). As with RNGs, this requirement mandates that the hashing function used in the other cryptographic operations be implemented in a cryptomodule.

The requirements below allow more than one cryptographic module to be used in providing the cryptographic functionality.

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.

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.1 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.Application Note: 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:

- 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.

a) 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).

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 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" refers 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 TSF 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 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_RIP.2 Full residual information protection

FDP_RIP.2.1 – **Refinement**: The TSF shall ensure that any previous information content of a resource is made unavailable upon the [selection: allocation of the resource to, deallocation of the resource from] all objects **or the TSF's completion of a function.**

Application Note: This SFR ensures residual biometric data (e.g., biometric samples stored temporarily in the capture device) is not available after its use in the functional component. This requirement was refined, since the resources may not be deallocated or reallocated (e.g., memory may be allocated to a function and never released). The intent is that once the TSF is has completed the processing of data, that data is no longer accessible. For example, clearing a biometric sample from the capture device memory after its operation, or from the "Matching and Comparison" component(s) after a match/no match decision is made.

5.1.4 Identification and Authentication (FIA)

FIA_AFL.1-NIAP-0425(1) Authentication failure handling (Against a single non-administrative user identifier)

FIA_AFL.1.1-NIAP-0425(1) – **Refinement:** The TSF shall detect when [a Security Administrator configurable number] **of** unsuccessful **biometric** authentication attempts occur related to [a claimed user identifier, **[selection: [assignment: other authentication mechanisms identified by the ST Author], none]**].

FIA_AFL.1.2-NIAP-0425(1) - **Refinement:** When the defined number of **consecutive** unsuccessful authentication attempts has been met, the TSF shall [ignore any further authentication attempts related to that user until the Security Administrator defined time period for non-administrative users has elapsed, or an action is taken by the Security Administrator].

Application Note: The intent of these requirements is to allow the Security Administrator to set the number of unsuccessful authentication attempts that are associated with a claimed user identifier that is **not** associated with an administrative role. The Security Administrator also has the option of configuring the TOE so further authentication attempts associated with the claimed user identifier are ignored until the Security Administrator takes an action (e.g., re-enables the account) or to ignore further authentication attempts associated with the user identifier until a Security Administrator configured time period for non-administrative users has elapsed (e.g., the TOE will not authenticate a user associated with that non-administrative claimed user identifier for 5 minutes). The ST author should fill in the selection if the TOE provides additional authentication mechanisms (e.g., multiple biometric authentication mechanisms, password mechanism). If the TOE reaches the Security Administrator configured setting, then an alarm is generated as required by FAU_SAA.1.

FIA_AFL.1-NIAP-0425(2) Authentication failure handling (Consecutive failed attempts)

FIA_AFL.1.1-NIAP-0425(2) - The TSF shall detect when [a Security Administrator configurable number] **of** unsuccessful authentication attempts occur related to [consecutive failed biometric authentication attempts].

FIA_AFL.1.2-NIAP-0425(2) – **Refinement:** When the defined number of **consecutive** unsuccessful authentication attempts has been met, the TSF shall [ignore any further authentication attempts from the offending capture device until the Security Administrator defined time period for consecutive failed authentication attempts has elapsed, or an action is taken by the Security Administrator].

Application Note: The intent of this requirement is to provide the Security Administrator the capability to set the number of consecutive failed authentication attempts, regardless of the claimed user identifier. This configurable number is different than that specified in FIA_AFL.1. For example, the Security Administrator may decide to set the failed number of authentication attempts against a non-administrative claimed user identifier to be three, and may set the failed number of consecutive failed authentication attempts to six. The Security Administrator defined time period is also distinct from the non-administrative user defined period defined in FIA_AFL.1(1). For example, the Security Administrator may set the time period for non-administrative users to be 5 minutes, but might configure the consecutive failed authentication attempts time period to be one hour. As with the pervious iteration, if the TOE reaches the Security Administrator configured setting, then an alarm is generated as required by FAU_SAA.1.

FIA_AFL.1-NIAP-0425(3) Authentication failure handling (Administrator Users)

FIA_AFL.1.1-NIAP-0425(3) - The TSF shall detect when [a Security Administrator configurable number] of unsuccessful authentication attempts occur related to [the administrator's account].

FIA_AFL.1.2-NIAP-0425(3) – **Refinement**: When the defined number of **consecutive** unsuccessful authentication attempts has been met, the TSF shall [ignore any further authentication attempts related to that user until the Security Administrator defined time period for administrative users has elapsed, or an action is taken by the Security Administrator].

Application Note: This iteration of FIA_AFL.1 applies to claimed user identifiers associated with an administrative role. The TOE has the ability to associate claimed user identifiers with administrative accounts, otherwise it would not know that that claimed user identifier may have to use the non-biometric authentication mechanism. The Security Administrator configurable number is distinct from the configurable number specified in the previous two iterations, as is the Security Administrator time period. This configurable setting applies to the any authentication mechanism used to authenticate administrative users of the TOE (e.g., biometric authentication mechanism(s), non-biometric authentication mechanism (e.g., password). As with the previous iterations of FIA_AFL.1, if the TOE reaches the Security Administrator configured setting, then an alarm is generated as required by FAU_SAA.1. Since the administrators may be required to use more than the biometric authentication mechanism, this requirement applies to any authentication mechanism used by the administrators.

FIA_ATD.1 User attribute definition

FIA_ATD.1.1 – **Refinement**: The TSF shall maintain the following list of security attributes belonging to **administrative** users:

- [trusted user identifier,
- role(s), and
- **[selection: [assignment: any other security attributes defined by the ST Author], none.]]**

and restrict the ability to assign and modify these security attributes to the Security Administrator.

Application Note: The TOE only associates security attributes with administrative users. An administrator may have more than one role associated with their trusted user identifier, however they can only act in one role at a time. Untrusted users do not have any interaction with the TOE that requires the association of security attributes. Due to the TOE having the ability to authenticate untrusted users that have not been enrolled on TOE, it may not be possible for the TOE to associate security attributes with untrusted users. The IT environment is responsible for associating security attributes with the user identifier authenticated via the TOE.

FIA_ENROLL_EXP.1 Enrollment

FIA_ENROLL_EXP.1.1 The TSF shall enforce the following rules:

a) Creation of the biometrics package, which contains:

- Trusted user identifier,
- reference template(s),
- [selection: [assignment: list of additional information determined by the ST Author], no additional information],

is performed during enrollment only;

b) A reference template cannot be modified;[6]

c) Enrollment (e.g., initial, refreshing reference templates, adding additional reference templates[7]) is performed by the Enrollment Administrator;

d) The failure-to-enroll rate is less than or equal to [assignment: rate assigned by ST Author that does not exceed a maximum value of 5%];

e) The Enrollment administrator is provided a quality metric of the newly created reference template;

f) Upon successful enrollment, the biometrics package is digitally signed by the TOE, and then encrypted before transfer to storage;

g) [selection: [assignment: other rules determined by the ST Author], none].

[6] The reference template cannot be modified once it has been created. For biometric technologies that continuously gather biometric characteristics to improve the quality of the reference template, a new template is created, rather than modifying an existing template.

[7] A biometric package may contain more than one reference template (e.g., a multifactor biometric device, to accommodate multiple vendors or technologies in a user's biometric package).

Application Note: The biometrics package may have more than one reference template associated with a trusted user identifier. This may be the case if the TOE that uses multiple biometric characteristics when authenticating a user (e.g., both thumb prints).

The assignment in item (a) may be filled in with other information such as which finger the user has enrolled with, a distress template (e.g., if the user attempts to authenticate with a biometric characteristic known to indicate a distress situation – using the right thumb instead of the left) or other information the TOE may use. If the ST author adds additional attributes, they should consider adding or augmenting existing requirements that use those attributes (e.g., adding a rule in FIA_UAU.5 that handles a distress indicator).

Item (b) ensures the reference template cannot be modified once it has been created. The TOE ensures the reference template cannot be modified while it is in the TOE's scope of control, and the TOE determines if the reference template has been modified while in storage or transit through the use of a cryptographic signature. In the case of "refreshing" a reference template, the old reference template is replaced with a new one, and the TOE must digitally resign the biometric package containing the new template.

Item (d) requires that the Enrollment Administrator be provided a quality metric of the newly created reference template. In a biometric system, the level of security achieved is known to be dependent on the quality of the biometric reference templates. If a poor enrollment is allowed, then that user may be open to easy attack by an imposter. This PP does not explicitly contain a minimally acceptable quality metric. This is left to the ST author and is discussed in the administrator guidance. The administrative guidance informs the Enrollment Administrator what are acceptable quality metrics. This allows the Enrollment Administrator to make an informed decision regarding the quality of the reference template and whether they should attempt to re-enroll the user.

For item (e), the ST author could add a rule that allows the TOE to be configured such that it will perform a comparison of any new reference template against the existing templates if they desire. This would allow the Enrollment Administrator the opportunity to find out which pairs of individuals cannot easily be distinguished by the product by performing inter-template comparisons. Such information should be kept confidential because an attacker may discover this information and try to make use of it.

FIA_SOS.1 Verification of secrets

FIA_SOS.1.1 The TSF shall provide a mechanism to verify that secrets meet [the following: For each attempt to use a non-biometric authentication mechanism, the probability that a random attempt to authenticate will succeed is less than one in 1×10^8].

Application Note: The ST specifies the method of authentication in FIA_UAU.5.1. When the non-biometric authentication is provided by a password mechanism, the ST shows that the restrictions upon passwords (length, alphabet, and other characteristics) result in a password space conforming to the specified metric. Administrators are able to select their authentication data (e.g., chose a password), but the TOE ensures that the chosen authentication data meets the identified metric.

FIA_SOS.2 TSF Generation of secrets

FIA_SOS.2.1 - The TSF shall provide a mechanism to generate secrets that meet [the following:

a) For each attempt to use the authentication mechanism, the False Acceptance Rate shall be in a Security Administrator settable range with a minimum value of: [assignment: rate assigned by ST Author] to a maximum value of: 1 in 100,000, and

b) False Rejection Rate shall be in a Security Administrator settable range with a minimum value of: [assignment: rate assigned by ST Author] to a maximum value of: 5 in 100.]

Application Note: In this TOE, the TSF generates the secret (i.e., the reference template) using an algorithm that is based on the biometric technology and uses a user's biometric characteristic. Since different biometric technologies provide varying degrees of False Acceptance Rates (FAR), this PP requires that at the maximum, the TOE will not have a FAR greater than 1 in 100,000. The ST author fills in the open assignment with a rate for a FAR their TOE can enforce. If the TOE cannot enforce a FAR less than 1 in 100,000 it is acceptable for the ST author to use the rate 1 in 100,000 in the assignment. Similarly, the False Rejection Rate (FRR) is specified as the maximum rate of false rejections the TOE will generate, and the ST author fills in the assignment with a rate that is less than or equal to the specified maximum rate of 5 in 100.

FIA_SOS.2.2 - The TSF shall be able to enforce the use of TSF generated secrets for [biometric authentication].

Application Note: The PP authors believe one aspect in ensuring that the TOE can enforce the rates specified in this requirement is the degree of quality of the reference templates. If the TOE allows a poor quality reference template to be accepted in the enrollment process, the belief is that these rates may be adversely affected.

FIA_UAU.2 User authentication before any action

FIA_UAU.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 Notes: This requirement really applies only to administrators, since they are the only users of the TOE that perform TSF mediated actions other than authentication. Non-administrative users perform no actions on the TOE other than requesting authentication, which is addressed by FIA_UAU_5.1.

FIA_UAU.5 Multiple authentication mechanisms

FIA_UAU.5.1 **Refinement**: The TSF shall provide [a biometric authentication mechanism, [assignment: non-biometric authentication mechanism that meets the strength of secrets metric defined in FIA_SOS.1], **[selection: [assignment: any other authentication mechanisms defined by the ST Author], none.]**] to **perform** user authentication.

Application Note: The TOE provides at a minimum, one biometric authentication mechanism and another non-biometric authentication mechanism (e.g., password mechanism, personal identification number). The non-biometric authentication mechanism is to be used, at the option of the Security Administrator, to authenticate administrators of the TOE. This non-biometric authentication mechanism satisfies the FIA_SOS.1 requirement.

The ST author may fill in the selection with an assignment of additional authentication mechanisms or may choose none in the selection. If the ST author fills in the assignment, then they should ensure that the additional mechanisms satisfy the appropriate FIA_SOS requirements, or iterate the FIA_SOS requirements to specify the strength of secrets those mechanisms provide. The ST author should also ensure that the rules in FIA_UAU.5.2 are enforced by the additional mechanisms, or create new rules that correspond to the behavior of the additional mechanisms.

If the TOE provides multiple biometric mechanisms, or multifactor authentication (biometric and non-biometric (e.g., token, password) mechanisms) for non-administrative users then the ST author should either iterate this requirement to accommodate additional authentication mechanisms, or specify the additional mechanisms and the rules that apply to those mechanisms. The TOE provides at least one biometric mechanism that satisfies the rules stated in this requirement. Any additional biometric mechanism(s) satisfy the rules specified by the ST author, which could be those specified in this requirement.

FIA_UAU.5.2 **Refinement**: The TSF shall authenticate any user's claimed identity according to the [following:

- For **non-administrative users**, the TSF shall authenticate a user and provide the IT environment with the trusted user identifier and a match/non-match decision according to the following rules:

    a) At the option of the Security Administrator a liveness check for [selection: involuntary response(s), voluntary response(s), vital sign(s), realness] which consists of [assignment: a description of what the TOE does in performing the liveness check] is performed and passed when the user supplied biometric characteristic is captured. If the liveness check fails, the TOE does not perform a comparison of the templates;

    b) verify the integrity of the biometrics package(s), confirming that the TOE has cryptographically signed the biometrics package, or the biometrics package has been cryptographically signed by a trusted authority;

c) in order to provide a match decision the comparison score is within the range specified by the maximum threshold and minimum threshold, otherwise a non-match decision is generated;

d) at the option of the Security Administrator, the TOE will not successfully authenticate the same claimed user identifier consecutively in a time duration specified by the Security Administrator;

e) [selection: [assignment: other rules determined by the ST Author], none].

Application Note: The ST author fills in the first selection based on what the TOE provides to the environment. If the TOE is used as an entry device on a door, the match/no match decision may be an electrical signal that opens the door if the TOE determines a match. If the TOE is providing authentication services to an IT environment, the expectation is the TOE will provide the IT environment with the user identifier that was supplied by the user, and the match/no match decision.

The selection in (a) is determined by the type of liveness check the TOE performs. This may consist of multiple instances of the identified types or some combination. The assignment in (a) provides a description of the technique the TOE uses in performing the liveness test. This is not intended to have the developer's proprietary algorithm described, rather it is intended to inform the end-user of what the TOE does with respect to liveness checking.

For item (b), the intent is that the TOE has the capability to maintain a list of trusted entities (e.g., Certificate Authorities) so the TOE can validate the integrity of a biometrics package that was not created by the TOE (e.g., the user was enrolled on another TOE) by using a trusted entity's public key to verify that entity had cryptographically signed the biometrics package.

For item (c) the TOE has a threshold range. The purpose of having a range is to provide the capability for a site to determine that if a match score is too high (e.g., a perfect match with the reference template) the TOE will provide a no match decision.

For item (d), the Security Administrator has the ability to configure the TOE to prevent the same user from successfully authenticating consecutively in a Security Administrator defined period of time. For example, the Security Administrator could configure the TOE so that once User X has successfully authenticated, User X cannot be the **next** user to be authenticated until 10 minutes have passed. This functionality is intended to ensure a user cannot attempt to "use" a residual left from a biometric characteristic from another user.

- For **administrative users**, the Security Administrator can choose that these users require authentication only by the biometric authentication mechanism(s), only by the non-biometric authentication mechanism, or both types of authentication mechanisms.

- When the TOE is configured to require administrators to use the biometric authentication mechanism, the TSF shall authenticate the administrative user and determine a match/non-match decision, according to the following rules:

  a) At the option of the Security Administrator a liveness check for [selection: involuntary response(s), voluntary response(s), vital sign(s), realness] which consists of [assignment: a description of what the TOE does in performing the liveness check] is performed and passed when the user supplied biometric characteristic is captured. If the liveness check fails, the TOE does not perform a comparison of the templates;

  b) verify the integrity of the biometrics package(s), confirming that the TOE has cryptographically signed the biometrics package, or the biometrics package has been cryptographically signed by a trusted authority;

  c) in order to provide a match decision the comparison score is within the range specified by the maximum threshold and minimum threshold, otherwise a non-match decision is generated;

  d) at the option of the Security Administrator, the TOE will not successfully authenticate the same claimed user identifier consecutively in a time duration specified by the Security Administrator;

  e) [selection: [assignment: other rules determined by the ST Author], none].

- When the TOE is configured to require administrators to use the non-biometric authentication mechanism, the TSF shall authenticate the administrative user according to the following rules:

  a) The authentication mechanism must provide a delay between failed authentication attempts, such that there can be no more than a Security Administrator configurable number of attempts per minute;

  b) Any feedback given during an attempt to use the authentication mechanism will not increase the probability of guessing above the metrics specified in FIA_SOS.1;

- When the TOE is configured to require administrators to use a biometric and non-biometric mechanism, the TSF shall authenticate the administrative user according to the following rules:

a) The rules for each mechanism specified for the administrator above hold true;

b) The administrator must be successfully authenticated by both mechanisms;

c) The authentication mechanisms provide no feedback unless both mechanisms are successful, other than to inform the user that the authentication process failed.

].

Application Note: The intent of item c) is to ensure the TOE does not indicate to the user which authentication mechanism failed (e.g., your password succeeded, but the biometric authentication failed).

FIA_UAU.7 Protected authentication feedback

FIA_UAU.7.1 – **Refinement**: The TSF shall provide only [instructional information] to **aid** the user **in supplying their biometric characteristic to the TOE**.

Application Note: This requirement means that the biometric system must not inform the user of any "score" against the threshold range that might help the attacker to fool the device in subsequent authentication attempts. Instructional information includes positioning information, volume, which finger to authenticate with, etc.

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.

FIA_USB.1-NIAP-0415 User-subject binding

FIA_USB.1.1-NIAP-0415: **Refinement:** The TSF shall associate the **following** user security attributes with subjects acting on behalf of that user: **[trusted user identifier, role, [selection: assignment: list of other security attributes determined by the ST Author to be bound, none]]**.

Application Note: As with FIA_UAU.2, the only users that attributes are associated with subjects are those of administrative users, since those are the only users that will have subjects with any "user" attributes.

5.1.5 Security Management Requirements (FMT)

FMT_MOF.1(1) Management of security functions behavior (audit selection)

FMT_MOF.1.1(1) - The TSF shall restrict the ability to *enable, disable, determine and modify the behavior of* the functions:

- [Security Audit (FAU_SEL)]

to [the Audit 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(2) Management of security functions behavior (audit review)

FMT_MOF.1.1(2) - The TSF shall restrict the ability to *enable, and modify the behavior of* the functions:

- [Security Audit (FAU_SAR)]

to [the Audit Administrator and Security Administrator].

Application Note: For the Audit review function, enable refers to the ability to use the audit review tool (e.g., start the program). "Modify the behavior" means the ability to set or unset the criteria used to select/search audit records for review.

FMT_MOF.1(3) Management of security functions behavior (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 Analysis (FAU_SAA); and
- 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. For FAU_ARP.1, behavior modification includes adjusting the defined time period that elapses before the TOE will resume performing authentication. For FAU_SAA, the intent of "modify the behavior" applies to the ability of the Security Administrator to selectively disable or enable specific events that may indicate a potential security violation.

FMT_MOF.1(4) - Management of security functions behavior (TSF non-Cryptographic Self-test)

FMT_MOF.1.1(4) - 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(5) - Management of security functions behavior (Cryptographic Self-test)

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

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

to [the Security Administrator].

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

FMT_MOF.1(6) Management of security functions behavior (Maintenance Mode)

FMT_MOF.1.1(6) - The TSF shall restrict the ability to *enable* the functions [to restore the TOE to a secure state from maintenance mode (FPT_RCV.2)] to [the Security Administrator].

Application Note: The intent of this requirement is if the TOE enters a state that it cannot automatically recover from and ensure that it will be able to enforce its security policies, that only a user acting in the role of the Security Administrator can restore the TOE to an operational state. One way this could be accomplished by requiring some form of a required password before startup from the maintenance mode state.

FMT_MOF.1(7) Management of security functions behavior (Enrollment)

FMT_MOF.1.1(7) – **Refinement:** The TSF shall restrict the ability to **perform**, *determine* and *modify the behavior of* the function [enrollment (FIA_ENROLL_EXP.1)] to [the Enrollment Administrator].

Application Notes: The Enrollment Administrator is the only user that is allowed to perform the enrollment function. "Determine the behavior" refers to the ability of the Enrollment Administrator to view any settings that the TOE may offer that affect the quality of the created reference template, as well as receiving the quality metric of the reference template when it is created. "Modify the behavior" refers to the Enrollment Administrator having the capability to set parameters that may affect the quality of the reference template when it is created, if the TOE offers such capability.

FMT_MOF.1(8) Management of security functions behavior (non-biometric Authentication Mechanism)

FMT_MOF.1.1(8) - The TSF shall restrict the ability to *enable and disable* the functions: [non-biometric authentication mechanism for required use on individual administrative roles] to [the Security Administrator].

Application Note: The Security Administrator has the ability to require the use of (enable or disable) the non-biometric authentication mechanism for individual administrative accounts.

FMT_MOF.1(9) Management of security functions behavior (Biometric Authentication Mechanism)

FMT_MOF.1.1(9) – **Refinement**: The TSF shall restrict the ability to *determine and modify the behavior of* the functions: [

- biometric authentication mechanism];

and to enable/disable the function:

- **biometric authentication mechanism for required use on individual administrative roles**

to [the Security Administrator].

Application Note: The Security Administrator has the ability to modify the behavior of biometric authentication mechanism by turning the liveness check on or off. Determine in this requirement applies to the Security Administrator being able to query the liveness check setting. The CC includes both the management (modifying the behavior) of a security function, and management of TSF data. It is sometimes confusing where to place certain aspects pertaining to the management of a TSF function, since managing TSF data can have an affect on the behavior of a TSF function. FMT_MTD.1(3) identifies TSF data that will have an impact on the behavior of this function and places restrictions on what administrative role can mange that TSF data.

The intent of this iteration and the previous iteration is to allow the Security Administrator the flexibility in determining which administrative accounts, at the granularity of an individual, require which authentication mechanisms are necessary for access. The intent is to allow an administrator to require specific individual administrative roles to authenticate to the TOE in a Security Administrator defined manner. The intent is to address a concern that the administrative role may not be able to administer the TOE if the biometrics authentication mechanism is rendered unavailable. This requirement, in conjunction with the previous iteration, could allow a Security Administrator to set up a **special** Security Administrator account that would require authentication only be the non-biometric account, while requiring a biometric authentication mechanism for all other administrator authentication attempts.

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

FMT_MTD.1.1(1) - The TSF shall restrict the ability to *modify* the [cryptographic security data] to [the Security Administrator].

Application Note: The intent of this requirement is to restrict the ability to configure the TOE's cryptographic policy to the Security Administrator. Configuring the cryptographic policy is related to things such as: setting modes of operation, key lifetimes, selecting a specific algorithm, manually entering keys, and key length.

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

FMT_MTD.1.1(2) – The TSF shall restrict the ability to *modify* the [time and date used to form the time stamps in FPT_STM.1] to [the Security Administrator].

FMT_MTD.1(3) Management of TSF data (Authentication Mechanism Data)

FMT_MTD.1.1(3) – **Refinement**: The TSF shall restrict the ability to *query, modify* the:

- [value of the minimum threshold; (FIA_UAU.5.2)
- value of the maximum threshold; (FIA_UAU.5.2)
- defined time period for blocking of further authentication attempts:
- time period for non-administrative users (FIA_AFL.1(1));
- time period for consecutive failed authentication attempts (FIA_AFL.1(2));
- time period for administrative users (FIA_AFL.1(3));
- defined time period has elapsed upon an alarm condition (FAU_ARP.1);
- Security Administrator configurable number of attempts per minute (FAU_UAU.5.2);
- time duration restricting the authentication of the same claimed user identifier consecutively;
- list of IT entities that the TOE will accept as cryptographic signing authorities;
- **[selection: [assignment: other data determined by the ST Author], none].**]

to [the Security Administrator].

FMT_MTD_EXP.1 Management of TSF data (Capture device unique identifier)

FMT_MTD_EXP.1.1 – The TSF [selection: provides unique capture device identifier(s), restricts the ability to query and modify the capture device identifier to the Security Administrator].

FMT_REV.1 Revocation

FMT_REV.1.1 - **Refinement:** The TSF shall restrict the ability to revoke security attributes associated with the **administrative** *users,* **[selection: [assignment: other additional resources specified by the ST Author], none]** within the TSC to [the Security Administrator].

FMT_REV.1.2 - The TSF shall enforce the rules:

- [revocation of a user's role is immediate; and
- [selection: [assignment: other rules as determined by the ST Author], none]].

Application Note: The security attributes associated with users are defined in FIA_ATD.1. If the ST author has added additional attributes in FIA_ATD.1 they should use the selection above to identify the rules for revoking those attributes.

FMT_SMR.2 Restrictions on security roles

FMT_SMR.2.1 – **Refinement**: The TSF shall maintain the roles:

- [Security Administrator;
- Audit Administrator;
- Enrollment Administrator, and
- **[selection: [assignment: any other roles determined by the ST Author], 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:
- the Audit Administrator and Security Administrator can review the audit trail; and
- any administrator 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 TSF (FPT)

FPT_ITT.1(1) Basic internal TSF data transfer protection (from disclosure)

FPT_ITT.1.1(1) **Refinement**: The TSF shall **use encryption to** protect TSF data from *disclosure* when it is transmitted between separate parts of the TOE.

FPT_ITT.1(2) Basic internal TSF data transfer protection (from undetected modification)

FPT_ITT.1.1(2) **Refinement**: The TSF shall **use a cryptographic digital signature to detect modification of** TSF data when it is transmitted between separate parts of the TOE.

FPT_ITC_EXP. 1 TSF confidentiality

FPT_ITC_EXP.1.1 - The TSF shall use encryption to protect the confidentiality of the biometrics package.

Application Note: This explicit requirement is necessary because the CC does not contain a requirement that specifies the desired functionality. The intent is that for this requirement and for FPT_ITT.1(1), the TOE encrypts the biometrics package using the cryptography specified in FCS_COP_EXP.2. The TOE can then send the biometrics package to a storage device that is not trusted over an unencrypted channel and ensure that the confidentiality of the data in the biometrics package is maintained. This requirement is different from the rest of the FPT_ITC family in that it does not require the storage device to be cryptography aware, it just stores the data it is provided.

The TOE may need to accept biometric packages that have been encrypted by another IT entity (i.e., user was enrolled on a different TOE). The key management requirements in

FCS_CKM._EXP.2 specify the acceptable means for accomplishing key establishment to allow the TOE to decrypt the biometrics package.

FPT_ITI_EXP.1 TSF detection of modification

FPT_ITI_EXP.1.1 The TSF shall use a cryptographic digital signature to detect modification of the biometrics package.

FPT_ITI_EXP.1.2 The TSF shall maintain a list of IT entities that it will accept as valid cryptographic signing authorities.

FPT_ITI_EXP.1.3 The TSF shall reject a biometrics package if modification is detected.

Application Note: This explicit requirement is necessary because the CC does not contain a requirement that specifies the desired functionality. The intent is that for this requirement and for FPT_ITT.1(2), the TOE cryptographically signs the biometrics package using the cryptographic digital signature algorithm specified in FCS_COP_EXP.3. The TOE maintains a list of trusted authorities and will accept biometrics packages that have been signed by those authorities. The key management requirements are presented in FCS_CKM._EXP.2 requirements and they specify the acceptable means for accomplishing key management.. There are no requirements for specifying the maintenance of the list of trusted authorities. This could be done locally by the Security Administrator, or remotely in a networked TOE. The TOE can send/receive a biometrics package to/from a storage device and not rely on the protection of the transmission medium, or the storage device to protect the integrity of the biometrics package. The TOE is able to verify the integrity of the biometrics package due to the fact that the biometrics package is cryptographically signed.

FPT_PHP_EXP.1 Detection of physical attack

FPT_PHP_EXP.1.1 The TSF shall detect physical tampering involving the following scenarios that might compromise the TSF: exposure of the internal components of biometrics TOE, [selection: [assignment: other scenarios determined by the ST author], none].

Application Note: This explicit requirement is necessary because the existing CC requirements do not allow for identifying the specific scenarios the TOE must detect.

This requirement includes all components of the TOE (e.g., capture device, enrollment device). If accomplished through the use of tamper-proof seals, the TOE developer provides the seals, or instructs the administrator how to obtain the appropriate seals. The administrator's guide instructs the administrator how and where to place the seals in order to allow detection of tampering. The intent of detect is that an audit record and alarm are generated.

FPT_PHP.3 Resistance to physical attack

FPT_PHP.3.1 **Refinement:** The TSF shall **react** [to the exposure of internal components] **of** the [biometrics TOE] by **zeroizing any cryptographic security parameters and** responding automatically such that the TSP is not violated.

FPT_RCV.2-NIAP-0406 Recovery from Failure

FPT_RCV.2.1-NIAP-0406 For [power failures], the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.2.2-NIAP-0406 When automated recovery from a failure or service discontinuity is not possible, the TSF shall enter a maintenance mode where the ability to return the TOE to a secure state is provided.

Application Note: The administrative guidance provides the Security Administrator with guidance/procedures that instruct them how to bring the TOE back into a secure state. If the TOE is unable to return to a secure state using automated procedures after a power failure the TOE enters a maintenance mode.

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. The ST author fills in the selection with any TSF data that is pertinent to their TOE (e.g., if the TOE provides more that one mode of operation, such as verification mode and identification mode, the mode of operation would go in the assignment).

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, although typically hash functions 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 the cryptographic 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 cryptomodule 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 Security Administrator has the ability to enable and disable this capability; this is specified in FMT_MOF.1(5). 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 TOE Access (FTA)

FTA_SSL.3 TSF-initiated termination

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

FTA_TAB.1 Default TOE access banners

FTA_TAB.1.1 - Refinement: Before establishing an **administrative** session, the TSF shall display an 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 of an administrator. The intent of this requirement is to advise administrators of warnings regarding the unauthorized use of the TOE. For untrusted users, the environment (IT or non-IT) would be responsible for displaying the appropriate banner.

FTA_TSE.1 TOE session establishment

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

5.2 TOE Security Assurance Requirements

The TOE assurance requirements for this PP do not map to a CC EAL. All assurance requirements are summarized in Table 5.4 below. The explicit requirements are 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.

Table 5.4 Assurance Requirements

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

Assurance Class	**Assurance Components**	
	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
	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 resistant

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 TOESF.

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 [all the requirements in the FIA Class contained in this PP] 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 Descriptive 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

Developer action elements:

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 all 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. While untrusted users will not typically have access to the TOE's interfaces (other than the capture device) this requirement also examines the ability for administrators to covertly obtain critical security parameters.

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 5. Additionally, this section describes the rationale for not satisfying all of the dependencies and the rationale for the strength of function (SOF) claim.

6.1 Rationale for TOE Security Objectives

Table 6.1 provides the mapping from Security Objectives to Threats and Policies, as well as a rationale that discusses how a threat or policy is addressed.

Table 6.1 Security Objectives to Threats and Policies Mappings

Threat/Policy	Objectives Addressing the Threat	Rationale
T.ADMIN_ERROR An administrator may incorrectly install or configure the 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_MULTIPLE_ROLE The TOE will provide multiple administrative roles to isolate non-overlapping administrative functions. 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 helps 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_MULTIPLE_ROLE 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 authenticate users by lowering the threshold. O.MANAGE also contributes to mitigating this threat by providing administrators the capability to view configuration settings. For example, if the Audit Administrator made a mistake when configuring the settings for events to be audited, providing them the capability to view the settings affords them the ability to review the settings and discover any mistakes that might have been made.
T.ADMIN_ROGUE An administrator's intentions may become malicious resulting in user or TSF data being compromised.	O.ADMIN_MULTIPLE_ROLE The TOE will provide multiple administrative roles to isolate non-overlapping administrative functions.	O.ADMIN_MULTIPLE_ROLE 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 impact the configuration settings affecting the TOE's authentication mechanism. On the other hand, if the Security Administrator becomes malicious they could affect the TOE's authentication mechanism, but the Audit Administrator may be able to detect those actions.
T.AUDIT_COMPROMISE	O.AUDIT_PROTECTION	O.AUDIT_PROTECT contributes to mitigating this threat by controlling access to the audit trail. No one

Threat/Policy	Objectives Addressing the Threat	Rationale
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.	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 or upon completion of a function that residual biometric data could not be reused. 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.	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 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 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.
T.BYPASS An attacker may bypass any component of the biometric product and gain unauthorized authentication.	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.SELF_PROTECTION mitigates this threat by requiring that the TOE respond to physical tampering in a manner that would not allow a user to authenticate or appear to be authenticated due to the bypassing of any component of the TOE. This objective also requires that the biometric data that is transmitted between physically separate components of the TOE be encrypted and cryptographically signed, which would prevent an attacker from "inserting" data in any communication path between TOE components.
T.CRYPT_ATTACK An attacker may defeat security functions through a cryptographic attack against the algorithm, through cryptanalysis on encrypted data, or through a brute-force attack and thereby gaining unauthorized authentication.	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.CRYPTOGRAPHIC_ FUNCTIONS The TOE shall provide cryptographic functions (i.e., encryption/decryption and digital signature operations) to maintain the confidentiality and allow for detection of modification of TSF data that is transmitted between physically separated portions of the TOE, or stored outside the TOE.	O.CRYPTOGRAPHY_VALIDATED contributes to mitigating this threat by requiring FIPS-approved functions to be used, thus lessening the chance that a poorly-thought-out algorithm could be compromised by an adversary. Additionally, the requirements levied on the cryptomodule by the FIPS process, and the verification of those requirements by the FIPS labs, helps add assurance that the cryptographic module can protect itself. O.CRYPTOGRAPHIC_ FUNCTIONS specifies the cryptographic algorithms and key management requirements that have been deemed appropriate to protect the confidentiality and integrity of data that is commensurate with the level of assurance provided by the TOE. This objective also requires that the cryptographic module providing these services has been validated to meet the FIPS 140-2 requirements as specified in this PP, which provides a moderate degree of assurance that the cryptographic algorithms, and key generation components are correctly implemented.
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,	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.RESIDUAL_INFORMATION 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

Threat/Policy	Objectives Addressing the Threat	Rationale
modified, or deleted), thus compromise the cryptographic mechanisms and the data protected by those mechanisms.	residual biometric data could not be reused. 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.	reallocated to another process. O.SELF_PROTECTION 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 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.HIGH_QUALITY_ARTIFACT An attacker may use a high quality artifact (e.g., artificial hand/fingerprint, life-size photograph, or other synthetic means) to gain unauthorized authentication.	O.AUTHENTICATION The TOE will provide a biometric authentication mechanism to authenticate users for the IT environment or non-IT environment.	In this context, forgery generally refers to the use of an artifact such that the biometric system is spoofed into accepting the artifact as coming from a live human being. It is not possible to make definitive statements on the potential for forging of biometric characteristics. Most biometric characteristics are not secret and may therefore be vulnerable to being copied. There will be varying degrees of difficulty involved. For example, it may be hard to copy a retinal pattern. This form of copying requires the use of a forgery to exploit the copy. Most biometric characteristics could, in principle, be forged given sufficient resources and justification. O.AUTHENTICATION addresses this threat by requiring a liveness test of the presented biometric characteristic. Due to the wide range of biometric technologies and the various forms of liveness tests associated with the different technologies, this PP requires that the ST Author state the type of liveness test done and what the test consists of. The intent is not that developers divulge their proprietary methods, but to provide end-users with enough information so they can intelligently compare what products perform in this regard and determine if that is suitable for their needs. The FAR number specified in this PP also contributes to mitigating this threat.
T.MIMIC An attacker may masquerade as an enrolled user by presenting their biometric characteristic that is similar, or by reproducing the biometric characteristics of the enrolled user (e.g., changing his/her voice, forging a signature, or other mean of mimicry) to gain unauthorized authentication.	O.AUTHENTICATION The TOE will provide a biometric authentication mechanism to authenticate users for the IT environment or non-IT environment. 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.	In some cases, an attacker may know that their biometric characteristics are very similar to those of an enrollee and attack that identity. This includes physical twins but is not confined to this case. The greater the number of enrollees, the more likely it is that the impostor resembles one of them. Some biometric products cannot distinguish between twins. Where the biometric product may confuse two individuals, an imposter may know which enrollees they best match and, for example, which finger to use. The risk is not confined to identical twins. In some cases, identical twins do not have identical biometric features (e.g. irises, fingerprints). In other cases, identical twins have identical biometric features (e.g. faces, DNA). As a result of FAR limitations, there may be pairs of unrelated individuals within relatively small samples, who can be reliably identified as each

Threat/Policy	Objectives Addressing the Threat	Rationale
		other. All behavioral biometrics are susceptible to mimic attacks. In a supervised environment, it is considerably more difficult to successfully mimic an enrollee without being detected. O.AUTHENTICATION addresses this threat by requiring a FAR of no more than 1 in 100,000. This threat cannot be totally mitigated and is an inherent weakness in some, if not all, of biometric technologies. O.ROBUST_TOE_ACCESS addresses this threat as it pertains to administrative accounts, since this objective requires the TOE to provide a non-biometric authentication mechanism to authenticate administrators if enabled by the Security Administrator.
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 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 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 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.
T.CORRUPTED_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	O.CHANGE_MANAGEMENT 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 requires that the developer's tools and techniques for implementing the design are documented. Having accurate and complete documentation, and having the

Threat/Policy	Objectives Addressing the Threat	Rationale
	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.	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 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 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.
T.POOR_ENROLLMENT An attacker may direct an attack against a low quality reference template and gain unauthorized authentication.	O.AUTHENTICATION The TOE will provide a biometric authentication mechanism to authenticate users for the IT environment or non-IT environment.	A low quality reference template can be caused by poor enrollment procedures, the quality of a user's biometric characteristic, or the biometric technology employed, that could lead to inferior biometric reference templates. O.AUTHENTICATION addresses this threat by requiring the TOE to provide the Enrollment Administrator a quality metric upon the enrollment of an individual. An acceptable quality metric will be dependent on the biometric technology and specific algorithms used by developers in their template generation and comparison function. Thus, a minimum quality metric is not specified in this PP. The administrative guidance documentation for the TOE will discuss quality metrics and what is acceptable for a specific TOE.
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 thereby causing potential security vulnerabilities.	O.CORRECT_ TSF_OPERATION The TOE will provide the capability to test the TSF to ensure the correct operation of the TSF at a customer's site. 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 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 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

Threat/Policy	Objectives Addressing the Threat	Rationale
		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 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_RESIDUAL_IMAGE An attacker may attempt to "reuse" an authorized user's biometric residual characteristic (e.g., finger print left on capture device) to gain unauthorized access.	O.AUTHENTICATION The TOE will provide a biometric authentication mechanism to authenticate users for the IT environment or non-IT environment.	O.AUTHENTICATION addresses this threat by requiring the TOE to provide the Security Administrator the option of disallowing the same user identifier to be authenticated in consecutive attempts. This threat is a concern to TOEs where a user comes into physical contact with the TOE's capture device (e.g., fingerprint). The rule in FIA_UAU.5.2 would prevent an attacker from using any residual biometric characteristic (e.g., a residual fingerprint left on the capture device) from being "re-used" subsequent to the legitimate user being authenticated.
T.RESIDUAL_DATA Residual biometric authentication data from a previous valid user if not cleared from memory may allow an attacker to gain unauthorized authentication.	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.RESIDUAL_INFORMATION counters this threat by ensuring that TSF data is not persistent when resources are released by one user/process and allocated to another user/process. The objective also ensures that the potential for residual data to be mistakenly reused is mitigated even though a process/subject has not deallocated assigned resources.
T. REFERENCE_TEMPLATE An attacker modifies or creates a biometric reference template in storage or transmission to/from storage to gain unauthorized authentication.	O.AUTHENTICATION The TOE will provide a biometric authentication mechanism to authenticate users for the IT environment or non-IT environment.	O.AUTHENTICATION counters this threat by providing the TOE the capability to verify that the biometric package has not been modified while it is in storage or during transmission to/from storage. This objective also ensures that a biometrics package has been created by the TOE, or another trusted entity through the enrollment process.
T.TAMPER An attacker may modify or otherwise alter the software or hardware components, the connections between them thereby gaining unauthorized authentication.	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.SELF_PROTECTION addresses this threat by ensuring that TOE provides a mechanism that detects the exposure of the internal TOE components, and by entering a state in which users could not gain unauthorized authentication. The TSF self-tests required by this objective ensure that the TOE's hardware is operating correctly and the software and TSF data have not been corrupted by means other than exposing the internal components (e.g., electromagnetic interference).
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.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	O.MANAGE 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 is necessary to mitigate this threat, because even if the security mechanisms do not allow an administrator to explicitly view TSF data, if TSF data were to inappropriately reside in a resource that was made

Threat/Policy	Objectives Addressing the Threat	Rationale
	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.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.	available to a administrator, that administrator would be able to inappropriately view the TSF data. O.SELF_PROTECTION 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 administrators could not view or modify TSF data or TSF executables that they are not authorized to access.
T.UNATTENDED_SESSION An attacker may gain unauthorized access to an administrator's 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 addresses this threat by requiring functionality in the TOE that places controls on administrative sessions. Administrative sessions are terminated after a Security Administrator defined time period of inactivity. Termination of an inactive administrative session reduces the risk of someone accessing the administrative console/workstation where the session was established, thus gaining unauthorized access to the session.
T.UNAUTHORIZED_ACCESS A user may gain access to administrative functions for which they are not authorized according to the TOE security policy.	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 addresses this threat by mandating the TOE provide a non-biometric authentication mechanism available for administrative user authentication. Using this mechanism eliminates some of the issues of relying solely on a biometric authentication mechanism. This objective also requires the TOE provide a mechanism that can be configured to allow administrative users to establish sessions only under certain circumstances (e.g., day, time, location). This feature helps mitigate when an attacker could establish an administrative session if they were able to obtain the administrators authentication data.
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 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) and generates an alarm once an event has occurred or a set threshold has been met. The method of alarm generation is left to the ST Author to describe since this PP attempts to include both networked and stand-alone TOEs the PP author's did not prescribe a method that might not be attainable by all types of TOEs.
T.UNKNOWN_STATE When the TOE is initially started or restarted after a failure, the security state of the TOE may 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 at a customer's site. O.SOUND_DESIGN	O.SOUND_DESIGN 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 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 power failure, the TOE attempts to

Threat/Policy	Objectives Addressing the Threat	Rationale
	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.	automatically restore itself to a secure state. For other types of failures this is not necessary. If the TOE cannot automatically recover from a power failure or experiences a different type of failure, the TOE enters a state that disallows further biometric authentication attempts and requires the Security Administrator to follow documented procedures to return the TOE to a secure state. O.CORRECT_TSF_OPERATION addresses this threat by ensuring that the TSF runs a suite of tests to successfully demonstrate 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 administrators 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 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 manner. The guidance also provides information about the corrective measures necessary when a failure occurs (i.e., how to bring the TOE back into a secure state).
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 satisfies this policy by ensuring that the TOE displays banner that provides administrators with a warning about the unauthorized use of the TOE. The displaying of the banner is not required for non-administrative users, since all TOEs may not have a display device capable of displaying a banner.
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.AUTHENTICATION The TOE will provide a biometric authentication mechanism to authenticate users for the IT environment or non-IT environment.	This policy has a somewhat different meaning in this TOE than other TOEs that allow untrusted users to process user data. Untrusted users are not provided access to the TOE other than providing their biometric characteristic to the capture device. O.AUDIT_GENERATION addresses this policy by providing the Audit 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 user identifier is recorded when any security relevant change is made to the TOE (e.g. modifying TSF data, start-stop of the audit mechanism). O.TIME_STAMPS plays a role in supporting this policy by requiring the TOE to provide a reliable time stamp (settable by only the Security Administrator). The audit mechanism is required to include the current date and time in each audit record. O.ROBUST_TOE_ACCESS supports this policy by requiring the TOE to identify and authenticate

Threat/Policy	Objectives Addressing the Threat	Rationale
		administrators prior to allowing any TOE access or any TOE mediated access on behalf of those users. This objective is necessary to counter this threat since O.AUTHENTICATION addresses the need for a biometrics authentication mechanism, and the Security administrator has the option of not requiring the use of the biometric authentication for administrative users. O.AUTHENTICATION is included since the biometric mechanism may be the only authentication mechanism required for administrators.
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 (i.e., encryption/decryption and digital signature operations) to maintain the confidentiality and allow for detection of modification of TSF data that is transmitted between physically separated portions of the TOE, or stored outside the TOE.	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 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.

6.2 Rationale for the Security objectives for the Environment

All of the security objectives for the environment 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.

6.3 Rationale for TOE Security Requirements

Table 6.2 maps the security functional requirements and the assurance requirements to the appropriate TOE objectives. A rationale is presented that provides the reader with a narrative of how the mapped requirement(s) are intended to satisfy the objective.

Table 6.2 - 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 threshold, and what quality metrics are acceptable when enrolling a user. 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 both administrators and non-administrators (e.g., password management guidelines). Since both administrative and non-administrative users of this TOE present their biometric characteristic to the TOE, this document would instruct all users how to correctly supply their characteristic. 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_MULTIPLE_ROLE	FMT_SMR.2	FMT_SMR.2 requires that three roles exist for administrative actions: the Security Administrator, who is responsible for

Objective	Requirements Addressing the Objective	Rationale
The TOE will provide multiple administrative roles to isolate non-overlapping administrative functions.		configuring the TOE's security policies; the Enrollment Administrator, who is restricted to enrolling users; and the Audit Administrator, who is restricted to reading and managing (e.g., backing up 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, with the exception of invoking self-tests and the ability of the Security Administrator to review audit, which was deemed necessary for them to perform trouble shooting or determining policy settings (e.g., getting a large number of failed attempts of authentication for a lot of users may require modification to the threshold setting). 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 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-NIAP-0415 FAU_SEL.1-NIAP-0407 FMT_MTD_EXP.1	FAU_GEN.1-NIAP-0410 defines the set of events that the TOE must be capable of recording. This requirement ensures that the Audit 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-410 ensures that the audit records associate a user identity with the auditable event. In the case of authenticated users, the association is accomplished with the user identifier. In the case of a failed authentication, the presented user identifier is associated with the event even though this identifier cannot be confirmed since these users are not authenticated. This is required since it may provide the Security Administrator with useful information (e.g., a specific user is targeted by an attacker). FAU_SEL.1-NIAP-0407 allows the Audit 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. FIA_USB.1 plays a role is satisfying this objective by requiring a binding of security attributes associated with users that are authenticated with the subjects that represent them in the TOE. This only applies to authenticated users, since the identity of unauthenticated users cannot be confirmed. Therefore, the audit trail may not always have the proper identity of the user that causes an audit record to be generated (e.g., an attacker/user providing another user's user identifier). FMT_MTD_EXP.1 requires that the TOE either provides capture devices with assured unique identifiers or provides the capability of setting identifiers to the Security Administrator (it is left to the Security Administrator to ensure they are unique). This requirement ensures that the audit trail indicates the capture device where the

Objective	Requirements Addressing the Objective	Rationale
		event occurred.
O.AUDIT_PROTECTION The TOE will provide the capability to protect audit information.	FAU_SAR.2 FAU_STG.1-NIAP-0423 FAU_STG.3 FAU_STG.NAIP-0414-1-NIAP-0429	FAU_SAR.2 restricts the ability to read the audit trail to the Security Administrator and Audit Administrator, 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 contributes to this objective by restricting the management of the audit trail to the Audit Administrator, this includes the backing up of the audit trail. This restriction helps ensure that audit records will not be lost and the Audit Administrator and Security Administrator will be able to associate events with users. FAU_STG.3 requires that an alarm is generated when the audit trail exceeds a capacity threshold established by the Audit Administrator. This ensures that the Audit Administrator has the opportunity to manage the audit trail before it becomes full and avoiding the possible loss of audit data. FAU_STG.NAIP-0414-1-NIAP-0429 allows the Audit 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 Audit Administrator) that would generate an audit record (e.g., depending on the FAU_SEL.1-NIAP-0407 configuration, users may no longer be allowed to authenticate) or the audit mechanism will overwrite the oldest audit records with new records (thus thwarting a denial of service attack). This requirement ensures that as a default, audit records will not be overwritten, and the Audit Administrator must select the overwrite option if that is what they desire.
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 FMT_MOF.1(3) FAU_SAR.3 FAU_SAR.1	FAU_SAA.1-NIAP-0407 defines the events that indicate a potential security violation and will generate an alarm. The triggers for the number of authentication failures are configurable by the Security Administrator. The failure of TSF self-tests, physical tampering, and detection of a modification of a biometrics package will generate an alarm. These events are independent of those selected for audit. For example if the Audit Administrator did not select the event of biometrics package modification in FAU_SEL, the Security Administrator could still configure the TOE to ensure that that event would generate an alarm. FAU_ARP.1 requires that the TOE generate an alarm when a potential security violation has been detected. Due to the wide range of TOE implementations, there is no specific requirement on how the alarm is to be generated. The ST author fills in the assignment of how their implementation will alert the administrator. FAU_SAR.1 provides the Audit Administrator and Security Administrator with the capability to read 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-410)

Objective	Requirements Addressing the Objective	Rationale
		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 search or sort the audit records residing in the audit trail. FAU_SAR.3 requires the administrators be able to establish the audit review criteria based on a user identifier, time and day, so that the actions of a user can be readily identified and analyzed. Allowing the administrators to perform searches or sort the audit records based on dates, times, subject identities provides the capability to extract the user activity to what is pertinent at that time in order facilitate the administrator's review. It is important to note that the intent of sorting in this requirement is to allow the administrators the capability to organize or group the records associated with a given criteria. FMT_MOF.1(3) restricts the ability to control the behavior of the audit and alarm mechanism to the Security Administrator. The Security Administrator is the only user that controls the behavior of the events that generate alarms and whether the alarm mechanism is enabled or disabled.
O.AUTHENTICATION The TOE will provide a biometric authentication mechanism to authenticate users for the IT environment or non-IT environment.	FIA_UAU.5 FIA_UID.2 FIA_ENROLL_EXP.1 FPT_ITC_EXP.1 FPT_ITI_EXP.1	FIA_UAU.5 requires the TOE to provide at least one biometrics authentication mechanism. This mechanism is the only mechanism that can authenticate non-administrative users and may be used at the discretion of the Security Administrator to authenticate administrative users. The rules regarding the use of the biometric authentication mechanism are specified in this requirement, including the circumstances under which the TOE provides a match/no match decision to the environment. Unlike an identification mode TOE, FIA_UID.2 requires that every user provide a user identifier before they are authenticated. This is essential for a verification mode biometrics device, and is one distinguishing factor from an identification mode biometrics device. Since a biometrics package is associated with a user identifier, it is essential to have a user supply their identifier before an authentication attempt can be made. FIA_ENROLL_EXP.1 is critical in establishing the requirements for the enrollment of a user. This requirement specifies what a biometrics package minimally consists of, and establishes the restrictions on the creation/modification of a biometric package (which includes the reference template). This requirement also mandates that the Enrollment Administrator be presented with a quality metric upon the potential enrollment of a user. The administrative guide discusses the enrollment procedure and how the quality metric affects the ability of the TOE to satisfy its FAR/FRR numbers. FPT_ITC_EXP.1 is necessary to ensure the confidentiality of the biometrics package is maintained when the package leaves the TOE's scope of control. Since the storage of the biometrics package is not required to be under the TOE's scope of control the storage device can be untrusted, yet the confidentiality of the package can be assured. While FPT_ITC_EXP.1 ensures the confidentiality of the biometrics package, FPT_ITI_EXP.1 is even more critical since it ensures the integrity of the biometrics package is maintained. This is necessary

Objective	Requirements Addressing the Objective	Rationale
		for the same reason that FPT_ITC_EXP.1 is necessary – the storage of the biometrics package is not trusted with respect to the TOE. If the integrity of the biometrics package cannot be assured, the authentication decision generated by the TOE cannot be trusted. One of the rules in FIA_UAU.5 requires that the integrity of the package be validated before a comparison of the reference template and live template can be made.
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. 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 at a customer's site.	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 (i.e., administrator) with the capability to ensure the TOE's security mechanisms continue to operate correctly in the field. FPT_TST_EXP.4 ensures end-user tests exist to demonstrate the correct operation of the security

Objective	Requirements Addressing the Objective	Rationale
		mechanisms required by the TOE that are provided by the hardware. 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. This requirement also validates the integrity 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. The FPT_TST_EXP.5 functional requirement has been added to address the critical nature and specific handling of the cryptographic keys. Since the cryptographic keys have specific FIPS PUB requirements associated with them it is important to ensure that any fielded testing on the integrity of these data maintains the same level of scrutiny as specified in the FCS functional requirements. This requirement allows the Security Administrator the option of having the cryptographic self-tests executed after the generation of every key. This may not be practical for some installations, therefore it is left to the Security Administrator's discretion.
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 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 (i.e., encryption/decryption and digital signature operations) to maintain the confidentiality and allow for detection of modification of TSF data that is transmitted between physically separated portions of the	FCS_CKM.1 FCS_CKM_SYM_EXP.1 FCS_CKM_ASYM_EXP.1	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

Objective	Requirements Addressing the Objective	Rationale
TOE, or stored outside the TOE.	FCS_CKM.4 FCS_COP_EXP.2 FCS_COP_EXP.3	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.12. 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 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 has been refined to apply only to administrative sessions, since an untrusted user does not establish a session with the TOE. In many cases the TOE may not have a display device and therefore no means of displaying a banner to untrusted users. It is expected that an administrator will have to have some type of display device to administrator the TOE (e.g., connect a console) and therefore a notice and consent banner is required.
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	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

Objective	Requirements Addressing the Objective	Rationale
	ATE_IND.2	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.2-NIAP-406	This objective is met by using the FPT_RCV.2-NIAP-406 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 (including the hardware tests required by FPT_TST_EXP.2.1) the TOE will enter a mode where it can no longer be assured of enforcing its security policies. Therefore, the TOE enters a state that disallows traffic flow and requires an administrator to follow documented procedures that instruct them on to return the TOE to a secure state. These procedures may include running diagnostics of the hardware, or utilities that may correct any integrity problems found with the TSF data or code. Solely specifying that the administrator reload and install the TOE software from scratch, while might be required in some cases, does not meet the intent of this requirement. An important aspect of this requirement is that upon a power failure, the TOE must attempt to automatically recover from the discontinuity. This aspect is included to eliminate the need of an administrator to have to "restart" every TOE under their purview due to a power failure at an installation.

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_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(7)
		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_MOF.1(1) specifies the ability of the Audit administrator to control the security function associated with audit generation. The ability to control this function has been assigned to the appropriate administrative roles. This requirement also allows the Audit Administrator to affect the events that are audited, turn audit off/on, and requires the capability exists that the Audit Administrator can determine/view the configuration settings.
		
		FMT_MOF.1(2) provides the Audit Administrator and Security Administrator the ability to enable a function to help them facilitate

Objective	Requirements Addressing the Objective	Rationale
	FMT_MOF.1(8) FMT_MOF.1(9) FMT_MTD.1(1) FMT_MTD.1(2) FMT_MTD.1(3) FMT_MTD_EXP.1 FMT_REV.1 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	the review of the audit trail. This requirement also ensures these administrative roles have the ability to select the event types (defined by the ST Author), and criteria that is required by this PP to enhance their ability to review the audit trail. This requirement limits the ability to perform these functions to only those users acting in the role of the Audit Administrator or Security Administrator. FMT_MOF.1(3) dictates the functionality required to manage the alarm functions of the TOE. The ability to control this function is limited to the Security Administrator and provides this role the capability of enabling or disabling the alarm function. This requirement also provides the Security Administrator with the capability to modify the behavior of the function that indicates a potential security violation. So as to ensure the mechanisms are configured as intended, the Security Administrator has the ability to view the conditions under which an alarm will be generated, and if alarm generation is enabled. FMT_MOF.1(4) The Security Administrator is the only role that is able to modify the behavior of the tests (e.g., select when they run, select a subset of the tests). This ensures that the self-tests will run no less than a frequency determined as necessary by the Security Administrator. This requirement also ensures the Security Administrator has the capability to determine that the behavior of the self-tests is configured as they intended. FMT_MOF.1(5) provides the capability for the security administrator to enable or disable the self testing of the cryptographic module after a key is generated. While the testing of the cryptographic components responsible for generating keys is important to ensure keys are generated correctly, it may be too resource consuming in some instances, and this management function provides the capability to turn the self tests off. FMT_MOF.1(6) was necessary to restrict the ability to restore the TOE to an operational mode after the TOE entered into a maintenance mode. The intent is to ensure that only the Security Administrator can restore the TOE, since this is the only administrative role that has the ability to view and configure the most critical configuration parameters. FMT_MOF.1(7) restricts the ability to enroll users to the Enrollment Administrator. Since correctly enrolling users is vital to the TOE's ability to correctly authenticate users, it was felt that it was appropriate that only users that understood the critical nature of enrollment, and were provided the necessary training would be allowed to perform enrollment. These users should be the only individuals assigned to the role of Enrollment Administrator. Since this TOE requires two authentication mechanisms (a biometric, and a non-biometric) that are to be administrated in different fashions, two management functions were deemed necessary. FMT_MOF.1(8) allows the Security Administrator to enable or disable the need for administrators to use the non-biometric authentication mechanism. FMT_MOF.1(9) provides capability to modify the behavior of the biometric authentication mechanism. This includes enabling or disabling of the liveness check, and setting the threshold that affects level of a match required in the comparison of the reference template and live template. This capability is also restricted to the Security

Objective	Requirements Addressing the Objective	Rationale
		Administrator role. FMT_MTD.1(1) is necessary to restrict the ability to modify the cryptographic security data that is used by the TOE to allow it to correctly enforce its security policy. A FIPS 140-2 module that is validated to provide a security level 3 for roles (as is required for this PP) ensures that those users that can manage the cryptographic module are uniquely identified individuals. This requirement levies a restriction, in that that individual most be assigned to the Security Administrative role as well. FMT_MTD.1(2) provides the capability of setting the date and time that is used to generate time stamps to the Security Administrator. These timestamps are critical since they are used in the audit trail to establish the sequence of events that have occurred in the TOE. It is important to allow this functionality, due to clock drift and other circumstances, but the capability must be restricted, and this requirement ensures only the Security Administrator can change the system time and date that generates this timestamp. Since the essence of a biometrics TOE is to perform authentication, FMT_MTD.1(3) ensures that only the Security Administrator has the flexibility to configure the TOE such that it behaves as required by their operational constraints. This requirement goes somewhat hand-in-hand with FMT_MOF.1(9). The CC includes both the management (modifying the behavior) of a security function, and management of TSF data. It is sometimes confusing where to place certain aspects pertaining to the management of a TSF function, since managing TSF data can have an affect on the behavior of a TSF function. FMT_MTD.1(3) identifies TSF data that will have an impact on the behavior of this function and places restrictions on what administrative role can mange that TSF data. This requirement identifies the TSF data the PP authors felt was essential in allowing a Security Administrator to manage the TOE. FMT_MTD_EXP.1 ensures that if the TOE provides the capability to set the identifier that the function is restricted to the Security Administrator. If the setting of the identifier is a provided capability, the TOE must also provide the Security Administrator with the capability to query the capture devices' identifiers. This capability provides the Security Administrator the ability to read all the capture device identifiers so that when they set an identifier they will know that it is unique. FMT_REV.1 ensures that the Security Administrator has the ability to revoke the assignment of a role to a specific user. This revocation is immediate and applies to all administrative roles identified in this PP. This helps a Security Administrator control what capabilities users have, if any, with respect to managing the TOE. FAU_SAR.1 ensures that the Audit and Security Administrators have the capability to review the audit records and that they are presented in a manner that is suitable for review (e.g., the administrators 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 Audit Administrator and Security Administrator. This capability exists for the Security to help facilitate any trouble shooting that they may have to perform.

Objective	Requirements Addressing the Objective	Rationale
		FAU_SAR.3 provides the Security Administrator and Audit Administrator 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 backup and 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 Audit Administrator the capability to establish a threshold of audit trail capacity, that when reached an alarm will be generated. If the audit trail becomes full FAU_STG.NIAP-0414-1-NIAP-0429 provides the Audit 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 Audit Administrator as an option to prevent a possible denial-of-service. FAU_SEL.1-NIAP-0407 provides the Audit 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.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.	FDP_RIP.2 FCS_CKM.4	FDP_RIP.2 is used to ensure the contents of resources are not available once the TSF is finished processing the TSF data, in addition to requiring that the data be made unavailable when reallocated to another subject. The requirement was refined since it is possible that the resource will not be deallocated or reallocated (e.g., memory assigned to a subject, never released and that memory would be used in subsequent authentication attempts. FCS_CKM.4 addresses this objective by ensuring the cryptographic keys are zeroized and are unavailable to unauthorized users.
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	AVA_SOF.1 FIA_AFL.1-NIAP-0425(1)-(3) FIA_ATD.1 FIA_SOS.1 FIA_SOS.2 FIA_UAU.2 FIA_UAU.5 FIA_UAU.7	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. A distinction between a verification mode and identification mode TOE is that the user must be identified and the comparison of the live biometric templates is done with the reference template associated with the user provided identity. While an attacker may continue attempting to authenticate by cycling through all the user identifiers (in essence manually performing what an identification mode TOE performs automatically). FIA_AFL is used to address this threat. In the context of this objective, the key is ensuring that an untrusted user cannot access an administrative account. FIA_AFL.1-NIAP-0425 has three iterations that provide a detection mechanism for unsuccessful authentication attempts for failed attempts against a single user identifier, consecutive failed attempts against any user identifiers, and failed attempts against an administrator account. For this objective, the third iteration is what plays a role in partially meeting the objective. The requirement

Objective	Requirements Addressing the Objective	Rationale
	FIA_UID.2 FTA_TSE.1 FTA_SSL.3	enables a Security Administrator settable threshold that prevents unauthorized users from gaining access to an administrators account 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. FIA_ATD.1 defines the attributes of users, including a user identifier 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 user identifier 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. This requirement ensures that untrusted users cannot be associated with a role and reduces the possibility of a user obtaining administrative privileges. 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. This TOE is somewhat unique in that it requires two authentication mechanisms, a biometric authentication mechanism and a non-biometric authentication mechanism for administrative access. The required use of these two authentication mechanisms is dictated at the option of the Security Administrator. If the Security Administrator desires, the non-biometric authentication is mandatory for administrative authentication. The FIA_SOS.1 requirement prescribes the metrics that must be satisfied when using this mechanism. The PP authors intentionally did not dictate that a password mechanism be required and allowed for other types of mechanisms (e.g. a PIN, Token). In any case, FIA_SOS.1 requires that the non-biometric authentication mechanism provide the ability for administrators to choose their "secret" in a space that cannot be guessed at random in less than probability of one in 1 x 10^8. It was thought that a PIN that consisted of 8 digits (0-9) could satisfy this requirement. Since this function is used solely for administrators, the intention is that administrators would be able to select their "secret" from this space. Since administrators may be responsible for administering a number of TOEs, it was deemed impractical to have the TOE generate the secrets and require the administrators to remember them. FIA_SOS.2 is directly related to the ability of the TOE to "generate" a secret based on a user's biometric characteristic. The PP authors believe that the TOE essentially generates a secret used to authenticate users based upon proprietary algorithms used by developers to generate a reference template and subsequent live templates for comparison. This authentication is optional, at the Security Administrator's discretion, for administrative users. The thinking is that if the capture device experience problems, the Security Administrator may want to have an account that can administer the TOE that does not rely on the biometric authentication mechanism. The PP authors struggled with trying to define a quality metric that they could impose on the TOE, but given the nature of the various technologies, it was felt that the FAR and FRR numbers would have to suffice in ensuring the TOE generates acceptable reference templates, which plays a significant role in the quality of

Objective	Requirements Addressing the Objective	Rationale
		the generated secret. The authors understand that the FAR and FRR numbers are dependent on other factors (e.g., the population of users enrolled, the quality of the biometric characteristic, the number of users enrolled), but this specification was felt the best that could be done at this time given the nature of biometric technologies and their application. FIA_UAU.2 simple requires that administrative users are authenticated before they perform any administrative actions. This is an unusual TOE, in that the only users of the TOE are administrative users. Untrusted users have no access to the resources resident in the TOE and have no interaction with the TOE other to authenticate themselves for access to a portal, or for possible mediation performed by another IT entity, therefore this requirement was refined to address only administrative users. FIA_UAU.5 provides the Security Administrator with the flexibility to determine the degree of authentication that is required of users that have access to the TOE itself (i.e. administrative users). This requirement provides the necessary rules for both biometric and non-biometric authentication mechanisms. The ability to configure the biometric authentication mechanism, and to require the use of the non-biometric authentication mechanism affords the Security Administrator the ability to dictate the degree of user authentication necessary to perform administrative activities. FIA_UAU.7 ensures that no feedback that affects their ability to circumvent the biometric authentication mechanism is presented to the user when they attempt to authenticate. The TOE is allowed to provide information that would allow the user to use the authentication mechanism in a correct manner (e.g., center your finger and press firmly, speak louder and slowly), but not provide information that may allow alteration to their presentation that would thwart the mechanism (e.g., you failed the liveness check, your comparison failed to pass the threshold by a factor of X). FTA_TSE.1 is used to control the ability of an administrator to establish a session with the TOE. The ability of a the Security Administrator to determine which users are able to administrate the TOE at a specific range of time, and from a specific location (this may only apply in a networked TOE) affords the TOE the ability to limit the exposure of the TOE to an attacker attempting to establish an administrative session. For example, if Security Administrator Joe, is only allowed to establish a session from 8-5, M-F, an attacker attempting to establish a session other than those hours would not succeed, regardless of them possessing the administrator's authentication data. FTA_SSL.3 contributes to satisfying this objective by limiting the exposure of an administrative session that is inactive for whatever reason. If an administrative session becomes inactive for a Security Administrator defined period, the session is terminated. This requirement applies both to remote and direct connections to the TOE.
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 FPT_ITT.1(1)	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

Objective	Requirements Addressing the Objective	Rationale
unauthorized disclosure.	FPT_ITT.1(2) FPT_TST_EXP.4 FPT_TST_EXP.5 FPT_PHP_EXP.1 FPT_PHP.3	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. While untrusted users are only intended to access this TOE via the capture device, this requirement ensures they cannot access other functionality provided by the TOE (i.e., administrative interfaces). This requirement also ensures that an administrator acting in a role only has access to the functions designated for that role. FPT_ITT.1(1) is necessary to satisfy this objective because it ensures that TSF data that is transmitted between components of the TOE is encrypted to prevent the disclosure of information. This data would include the live template as it leaves the capture device, or as it is transmitted between other parts of the TOE. This would also include any TSF data that is sent from an administrative console to the TOE if that console is "networked" with the TOE. This would not apply to TSF data that is configured from a console that is connected via a communication path (e.g., serial cable, USB port) that ensures the data cannot be disclosed. The disclosure of TSF data could create an opportunity for the TOE to be rendered ineffective in enforcing its security policies. FPT_ITT.1(2) ensures the integrity of the TSF data is maintained as it is transmitted between various parts of the TOE. Ensuring the integrity of the TSF data is crucial in order to ensure the TSF can enforce its security policies. FPT_TST_EXP.4 provides capability for the administrators to ensure that the TSF hardware is operating correctly, and that the resident TSF data and TSF software have not been corrupted. This aspect is critical in the administrator's determination that the TSF can indeed protect itself, or of the fact that something has happened to bring into question the TSF's ability to protect itself. FPT_TST_EXP.5 is used to ensure that the components used in generating cryptographic keys are working correctly. Since cryptography plays an important role in the TSF's ability to enforce security policy, this requirement contributes significantly to this objective. FPT_PHP_EXP.1 plays a diminished role in satisfying this objective in that it can generate an alarm and audit record notifying the Security Administrator and Audit Administrator that a potential physical attack has been mounted against the TOE. This notification affords the administrators the opportunity to inspect the TOE and determine if the TOE has been physically compromised. FPT_PHP.3 goes one step further than FPT_PHP_EXP.1 since it causes the TOE to ensure that if it is physically compromised that the security policies cannot be circumvented. FPT_PHP.3 is important, since due to the nature of biometric TOE installations it might not be

Objective	Requirements Addressing the Objective	Rationale
		possible to react to a physical attack even given a notification as required by FPT_PHP_EXP.1.
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, the interface consists of the interface presented to the untrusted user (i.e., the capture device), as well as the interface presented to administrators (e.g., administrative commands). If the TOE provides a network interface, a specification of the network interface (including the network interface hardware) 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

Objective	Requirements Addressing the Objective	Rationale
		ADV_INT_EXP.1 requirement, the evaluators will ensure that the 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(2)	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(2) satisfies the rest of this objective by providing the capability to set the time used for generating time stamps to the Security Administrator. This functionality allows the Security Administrator to ensure the time and date are correctly set, while restricting this function from unauthorized use.
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.

6.4 Rationale for Assurance Requirements

The assurance selection was based on:

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

The EAL definitions and assurance requirements in Part 3 of the CC were reviewed and the *Medium Robustness Assurance Package* as defined in Section 5.2 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. 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 6.2.

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 6.3 identifies the functional requirement, its correspondent dependency and the analysis and rationale for not supporting the CC defined dependency in this PP.

Table 6.3 - Broken Dependency Rationale

Requirement	Dependency	Dependency Analysis and Rationale
FIA_UAU.1 FIA_UAU.2 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_MTD.1 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.
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 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_SMF.1	The requirements FMT_MOF.1, and FMT_MTD.1 express the functionality

Requirement	Dependency	Dependency Analysis and Rationale
FMT_MTD.1		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.

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. However, the PP authors felt that the FIA_SOS.1 and FIA_SOS.2 requirements were necessary to levy specific criteria on the strength of both the non-biometric and biometric authentication mechanisms. FIA_UAU.5 also has a strength of function aspect, as this requirement is where the live template and the reference template are compared and the resulting decision is based on a probability of a match.

6.7 Rationale for Explicit requirements

The rationale for the inclusion of the explicit requirements found in this PP is presented in Table 6.4. Due to the unique nature of biometric technologies the PP authors found it necessary to write explicit requirements since the existing CC requirement did not capture the security functionality desired.

Table 6.4 Rationale for Explicit Requirements

Explicit Requirement	Identifier	Rationale
FAU_ENROLL_EXP.1	Enrollment	This requirement is necessary because the CC does not contain an SFR that addresses the desired security functionality required for the enrollment of a user in a biometrics TOE. This requirement specifically states what is minimally required in a biometrics package and the constraints regarding access and modification of the biometrics package.
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 or how it is to be validated other than a CC evaluation facility.
FCS_CKM_SYM_EXP.1	Cryptographic Key Establishment for AES symmetric keys	This two explicit requirements is 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)	These explicit FCS_COP requirements were created due to the nature of requiring FIPS validation and the relationship between the requirements that could not be clearly captured in the existing FCS_COP CC requirements.
FCS_COP_EXP.3	Cryptographic Operation (Digital Signature Generation/Verification)	
FCS_COP_EXP.5	Cryptographic Operation (Random Number Generation)	
FCS_COP_EXP.6	Cryptographic Operation (Cryptographic Hashing Function)	

Explicit Requirement	Identifier	Rationale
FMT_MTD_EXP.1	Management of TSF data (Capture device unique identifier)	This explicit requirement is necessary because the PP authors did not want to require the TOE to provide the capability to query and set the capture device identifier if the TOE developer uses some means to ensure the capture device identifiers are unique (e.g., serial number). The CC does not contain an existing requirement that captures the intent of this explicit requirement.
FPT_ITC_EXP. 1	TSF confidentiality	This explicit requirement is necessary because the CC does not contain a requirement that specifies the desired functionality. The PP authors did not want to require the storage device be aware of the cryptography used or to have the cryptographic keys to decrypt the data.
FPT_ITI_EXP.1	TSF detection of modification	This explicit requirement is necessary because the CC does not contain a requirement that specifies the desired functionality. The PP authors did not want to require the storage device be aware of the cryptography used or to have the cryptographic keys to sign the biometric package.
FPT_PHP_EXP.1	Detection of physical attack	This explicit requirement is necessary because the existing CC requirements do not allow for identifying the specific scenarios the TOE must detect.
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 is 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	

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.

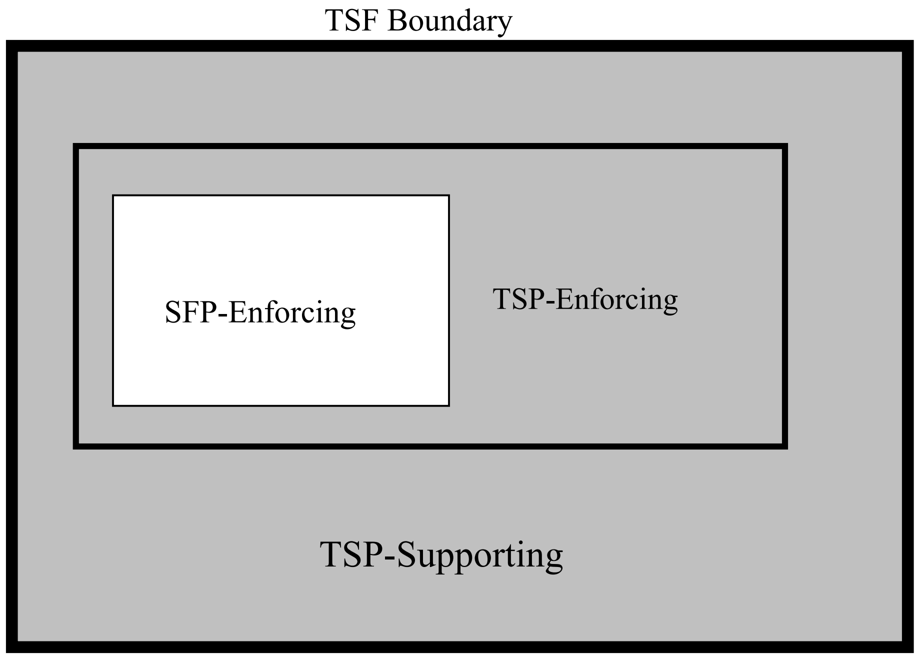


Figure AA. SFP-enforcing may only be a subset of TSP-enforcing functions.

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).

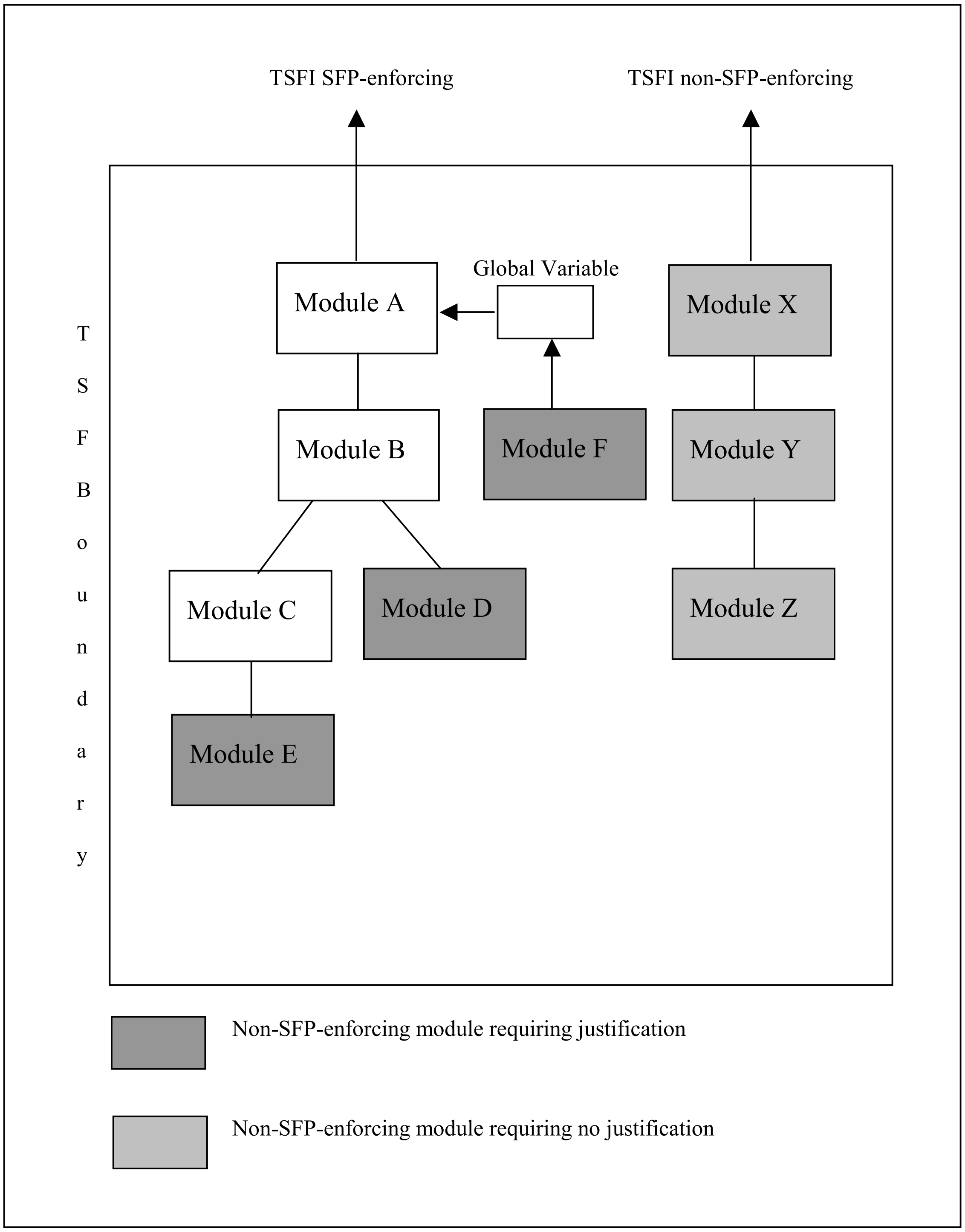


Figure XX. Example of non-SFP-enforcing modules requiring justification.

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.[8]

- 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:

---

[8] *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.*

- The number of modules that modify a global variable: In general, only a single module should be allocated the responsibility for controlling the contents of a global variable, but there may be situations in which a second module may share that responsibility; in such a case, sufficient justification must be provided. It is unacceptable for this responsibility to be shared by more than two modules. (In making this assessment, care should be given to determining the module actually responsible for the contents of the variable; for example, if a single routine is used to modify the variable, but that routine simply performs the modification requested by its caller, it is the calling module that is responsible, and there may be more than one such module). Further, as part of the complexity determination, if two modules are responsible for the contents of a global variable, there should be clear indications of how the modifications are coordinated between them.

- The number of modules that reference a global variable: Although there is generally no limit on the number of modules that reference a global variable, cases in which many modules make such a reference should be examined for validity and necessity.

- *content:* two modules are content coupled if one can make direct reference to the internals of the other (e.g. modifying code of, or referencing labels internal to, the other module). The result is that some or all of the content of one module are 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.

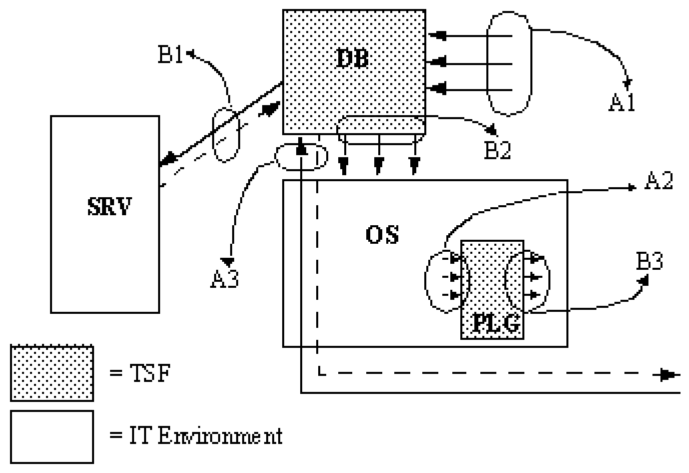
In order to identify the software interfaces to the TSF, the parts of the TOE that make up the TSF must be identified. This identification is formally a part of ADV_HLD_EXP analysis. In this analysis, a portion of the TOE is considered to be in the TSF under two conditions:

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, no firewall-provided services such as telnet). 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.[9] 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.

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.

[9] *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."*

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) *BioAPI Specification,* Version 1.1, March 16, 2001.

3) *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.*

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

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

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

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

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

9.0 TERMINOLOGY

9.1 Specific Biometrics Terminology

Attack -- An act attempting to violate the security policy of an IT system.

Attacker - An attacker is any individual who is attempting to subvert the operation of the biometric system. The intention may be either to subsequently gain illegal entry to the portal or to deny entry to legitimate users.

Attempt – The submission of a biometric sample to a biometric system for identification or verification.

Authentication/Authenticate, Biometric – The biometric process of either identifying or verifying a user.

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 a Target of Evaluation Security Policy, perform an operation.

Best Match – The biometric presented is not 100% exactly the same as the reference user template but is the closest match.

Biometric – Measurable physical characteristic or personal behavioral trait used to recognize the identity or verify the claimed identity of an individual.

Biometric Data – The extracted information taken from the biometric sample and used either to build a reference template or to compare against a previously created reference template.

Biometric Package – Record created by the biometric TOE that cryptographically bind the user's identity and additional information with the biometric template for storage.

Biometric Raw Data -- The initial data from a biometric sensor device from which a biometric template is derived.

Biometric Record -- The biometric raw data, biometric sample, and/or the biometric template of an individual.

Biometric Sample – Data representing a biometric characteristic of a user as captured by a biometric system.

Biometric System – An automated system capable of capturing a biometric sample from a user, extracting biometric data from that sample, comparing the biometric data with that contained in one or more reference templates, deciding how well they match, and indicating whether or not an authentication of identity has been achieved.

Capture – The process of taking a biometric sample from the user.

Claimed user identifier - The name or index of a claimed user identity, used by a biometric system for verification.

Comparison – The process of comparing biometric data with a previously stored reference template or templates.

Enrollee – A person who has a biometric reference template stored in a biometric package.

Enrollment – The process of collecting biometric samples from a user and the subsequent preparation, encryption, and storage of biometric reference templates representing that person's identity.

Exact Match – The biometric presented is 100% exactly the same as the reference user template.

Failure to Acquire -- Failure of a biometric system to capture and extract biometric data.

Failure to Acquire Rate -- The frequency of a failure to acquire.

Failure-to-Enroll – Any irrecoverable failure in the enrollment process.

Failure-to-Enroll Rate - The probability that a biometric system will have a failure-to-enroll.

False Acceptance – When a biometric system incorrectly identifies an individual or incorrectly authenticates an impostor against a claimed identity.

False Acceptance Rate (FAR) – The probability that a biometric system will incorrectly identify an individual or will fail to reject an imposter. It is stated as follows:

FAR = NFA/NIIA or FAR=NFA/NIVA

Where **FAR** is the false acceptance rate

Where **NFA** is the number of false acceptances

Where **NIIA** is the number of imposter identification attempts

Where **NIVA** is the number of imposter verification attempts

False Rejection – When a biometric system fails to identify an enrollee or fails to verify the legitimate claimed identity of an enrollee.

False Rejection Rate (FRR) – The probability that a biometric system will fail to identify an enrollee, or verify the legitimate claimed identity of an enrollee. It is stated as follows:

FRR=NFR/NEIA or **FRR=NFR/NEVA**

Where **FRR** is the false rejection rate

Where **NFR** is the number of false rejections

Where **NEIA** is the number of enrollee identification attempts

Where **NEVA** is the number of enrollee verification attempts

Identification/Identify, Biometric – The one-to-many process of comparing a submitted biometric sample against all of the biometric reference templates on file to determine whether it matches any of the templates and, if so, the identity of the enrollee whose template was matched. The biometric system using the one-to-many approach is seeking to find an identity amongst a database rather than authenticate a claimed identity. Contrast with "Authentication".

Identity -- A representation (e.g., a string) uniquely identifying an authorized user.

Imposter – A person who submits a biometric sample in either an intentional or inadvertent attempt to pass him/herself off as another person who is a legitimate enrollee.

Match Score – A numeric value or set of values derived from the comparison by the biometric system of a biometric sample with a template.

Matching -- The process of comparing a biometric sample against a previously stored template and scoring the level of similarity.

Portal – The logical or physical point beyond which the protected assets reside. For example, a physical portal may be the locking mechanism on a door. A logical portal may be an authentication measure taken prior to gaining access to a computer.

Physical/Physiological Biometric – A biometric that is characterized by a physical characteristic rather than a behavioral trait.

Replay attack – An attack in which a valid data transmission is maliciously or fraudulently repeated, either by the originator or by an adversary who intercepts the data and retransmits it, possibly as part of an imposter attack.

Secure State – A condition of normalcy, which occurs when all functions operate securely, as designed.

Template – Data that represents the biometric measurement of an enrollee, used by a biometric system for comparison against subsequently submitted biometric samples.

Threshold – The acceptance or rejection of biometric data is dependent on the match score falling above or below a defined limit. The threshold may be adjustable so that the biometric system can be more or less strict, depending on the requirements of any given biometric application.

Trusted user identifier – The name or index of a user identity that is derived from a trusted source.

User -- Any entity (human user or external IT entity) outside a Target of Evaluation that interacts with the Target of Evaluation.

Verification, Biometric – The one-to-one process of comparing a submitted biometric sample against the biometric reference template of a single enrollee whose identity is being claimed, to determine whether it matches the enrollee's template. Contrast with Biometric "Identification".

Zero Effort Forgery – An arbitrary attack on a specific enrollee identity in which the imposter masquerades as the claimed enrollee using his or her own biometric sample.

9.2 Common Protection Profile Terminology

In the Common Criteria, many terms are defined in Section 2.3 of Part 1. The following are a definitions of terms some of which are used in this PP, and are 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[10] and the disclosure and modification of data.[11]

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.

[10] Hardware and software.

[11] Stored or communicated.

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.

Availability -- Timely[12], 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,

[12] According to a defined metric.

- 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 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. The TOE uses integrity labels 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.[13]

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 -- An object that exhibits all of the following characteristics:

- The object may be used to transfer information between subjects of differing user identities within the TSF.
- Subjects in the TOE must be able to request a specific instance of the object.
- The name used to refer to a specific instance of the object must exist in a context that potentially allows subjects with different user identities to request the same instance of the object.

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.

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

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. ADV_INT_EXP.1
- **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
CC	Common Criteria for Information Technology Security Evaluation
DoD	Department of Defense
EAL	Evaluation Assurance Level
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
IT	Information Technology
MRE	Medium Robustness Environment
NIAP	National Information Assurance Partnership
NIST	National Institute of Standards and Technology
NSA	National Security Agency
PKI	Public Key Infrastructure
PP	Protection Profile
RNG	Random Number Generator
SFP	Security Function Policy
SOF	Strength of Function
ST	Security Target
TOE	Target of Evaluation
TSE	TOE Security Environment
TSF	TOE Security Function
TSP	TOE Security Policy

11.0 REFINEMENTS

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

11.1.1 Security Audit (FAU)

FAU_ARP.1 Security alarms

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:

1. local console,
2. remote administrator sessions that exist, and;
3. remote administrator sessions that are initiated before the alarm has been acknowledged, and;
4. at the option of the Security Administrator, generate an audible alarm, and;
5. [assignment: other methods determined by the ST author]]

upon detection of a potential security violation.

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 **(for the [selection] level of audit)** listed in Table 5.3; 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 (if applicable); and

b) For each audit event type **(time)**, based on the auditable event definitions of the functional components included in the PP/ST, [information specified in column three in Table 5.3].

FAU_SAA.1-NIAP-0407 Potential violation analysis

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

a) Accumulation **(or combination)** of [

- a Security Administrator specified number of authentication failures against a single non-administrative user identifier,

- a Security Administrator specified number of consecutive failed authentication attempts,

- a Security Administrator specified number of authentication failures against an administrative user identifier;

b) Any failure of the cryptographic self-tests;

c) Any failure of the other TSF self-tests;

d) Any failure to generate a cryptographic key;

e) Detection of physical attack;

f) Any failure to decrypt a biometrics package;

g) Detection of modification of a biometrics package] **(known to indicate a potential security violation)**;

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

FAU_SAR.1 Audit review

FAU_SAR.1.2 – **Refinement**: The TSF shall provide the audit records in a manner suitable for the **(user)** Audit Administrator and Security Administrator to interpret the information.

FAU_SAR.2 Restricted audit review

FAU_SAR.2.1 – **Refinement**: The TSF shall prohibit all users read access to the audit records, except **(those users that have been granted explicit read-access)** the Audit Administrator and Security Administrator.

FAU_SEL.1-NIAP-0407 Selective Audit

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

a) [user identifier;

b) event type;

c) success of auditable events;

d) failure of auditable events; and

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

FAU_STG.1-NIAP-0423 Protected audit trail storage

FAU_STG.1.1-NIAP-0423 – **Refinement**: The TSF shall **(protect)** restrict the backup and 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 **(unauthorised)** 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 **(take)** [generate an alarm by [assignment: method determined by the ST Author to generate the alarm]], if the audit trail exceeds [an Audit Administrator settable percentage of storage capacity].

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 Audit Administrator the capability to select one **(or more)** of the following actions: prevent auditable events, except those taken by the Audit Administrator, overwrite the oldest stored audit records or [selection: [assignment: other actions to be taken in case of audit storage failure], no other actions] **(and)** to be taken if the audit trail is full.

FAU_STG.NIAP-0414-1.2-NIAP-0429 **Refinement**: The TSF shall as a default [prevent auditable events, except those taken by the Audit Administrator] if the audit trail is full **(and no other action has been selected)**.

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

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)** for all key sizes **([assignment])** that meet **(the following:)** [one of the standards defined in Annex C to FIPS 140-2].

FCS_CKM.4 Cryptographic Key Destruction

FCS_CKM.4.1: **Refinement**: The TSF shall destroy cryptographic keys in accordance with a cryptographic key **(destruction)** zeroization method **([assignment])** 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].

FIA_AFL.1-NIAP-0425(1) Authentication failure handling (Against a single non-administrative user identifier)

FIA_AFL.1.2-NIAP-0425(1) - **Refinement**: When the defined number of consecutive unsuccessful authentication attempts has been met **(or surpassed)**, the TSF shall [ignore any further authentication attempts related to that user until the Security Administrator defined time period for non-administrative users has elapsed, or an action is taken by the Security Administrator].

FIA_AFL.1-NIAP-0425(2) Authentication failure handling (Consecutive failed attempts)

FIA_AFL.1.2-NIAP-0425(2) - **Refinement**: When the defined number of consecutive unsuccessful authentication attempts has been met **(or surpassed)**, the TSF shall [ignore any further authentication attempts related to that user until the Security Administrator defined time period for non-administrative users has elapsed, or an action is taken by the Security Administrator].

FIA_AFL.1-NIAP-0425(3) Authentication failure handling (Administrator Users)

FIA_AFL.1.2-NIAP-0425(3) – **Refinement**: When the defined number of consecutive unsuccessful authentication attempts has been met **(or surpassed)**, the TSF shall [ignore any further authentication attempts related to that user until the Security Administrator defined time period for non-administrative users has elapsed, or an action is taken by the Security Administrator].

FIA_ATD.1 User attribute definition

FIA_ATD.1.1 – **Refinement**: The TSF shall maintain the following list of security attributes belonging to **(individual)** administrative users:

- [trusted user identifier,
- role(s), and
- [selection: [assignment: any other security attributes defined by the ST Author], none.]]

and restrict the ability to assign and modify these security attributes to the Security Administrator.

FIA_UAU.5 Multiple authentication mechanisms

FIA_UAU.5.1 **Refinement**: The TSF shall provide [a biometric authentication mechanism, [assignment: non-biometric authentication mechanism that meets the strength of secrets metric defined in FIA_SOS.1], [selection: [assignment: any other authentication mechanisms defined by the ST Author], none.]] to **(support)** perform user authentication.

FIA_UAU.7 Protected authentication feedback

FIA_UAU.7.1 – **Refinement**: The TSF shall provide only [instructional information] to aid the user **(while the authentication is in progress)** in supplying their biometric characteristic to the TOE.

FPT_ITT.1(2) Basic internal TSF data transfer protection (from undetected modification)

FPT_ITT.1.1(2) **Refinement**: The TSF shall **(protect)** use a cryptographic digital signature to detect modification of TSF data **(from [selection])** when it is transmitted between separate parts of the TOE.

FPT_PHP.3 Resistance to physical attack

FPT_PHP.3.1 **Refinement**: The TSF shall **(resist)** react [to the exposure of internal components] **(to)** of the [biometrics TOE] by responding automatically such that the TSP is not violated.

FPT_SEP.2 SFP domain separation

FPT_SEP.2.3 - **Refinement**: The TSF shall maintain the part of the TSF related to [cryptography] in **(a security domain)** 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.3 TSF-initiated termination

FTA_SSL.3.1 - **Refinement**: The TSF shall terminate **(a user)** an administrative session after a [Security Administrator-configurable time interval of session inactivity].

FTA_TAB.1 Default TOE access banners

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