The PKI Secure Kernel Protection Profile

PKIPROtection
file

Version “1.1 evaluated”

April 4, 2002

This document is the first of a series of Protection Profiles to cover a “Public Key Infrastructure” architecture. It has been produced by the so called “*PKI PP working group*”.

The authors gracefully donate this document to the public domain, encouraging its wide distribution, usage and adoption. No patent of copyright shall be claimed by the group, who, by putting it in public domain, deter others of doing so.

The “*PKI PP working group*” at the time of issuing this version is defined by the following members. The actual group members can be reached at `pkipp@safelayer.com`.

```
"Arturo Ribagorda Garnacho" <arturo@inf.uc3m.es>
"Benjamín Ramos" <benja1@inf.uc3m.es>
"Eduardo Rojas" <erojas@dissc.presidencia.gob.es>
"Fernando Fazio" <ffazio@setsi.mcyt.es>
"Fernando Ledrado" <fernando.ledrado@comadrid.es>
"Fernando Piera Gómez" <fpiera@indra.es>
"Francisco Jordán" <jordan@safelayer.com>
"Francisco López" <Francisco.Lopez-Crespo@sgci.dgopti.map.es>
"Gemma Déler" <GDeler@lgai.es>
"Jaime Agudo" <jagudo@elacaixa.com>
"Jaime Pereda" <jaime.pereda@amena.es>
"Jaume Díaz Serret" <jdiaz@ctele.gencat.es>
"Javier Balsa" <javier.balsa@comadrid.es>
"Javier Santos" <fsantos@europamc.com>
"Jordi Buch" <jbt@safelayer.com>
"Jordi Iñigo" <jig@safelayer.com>
"Jorge Dávila Muro" <jdavila@fi.upm.es>
"José A. Mañas" <jmanas@dit.upm.es>
"José C. Santos" <jcsantos@exceldata.es>
"José Luis Huertas" <huertasjl@inta.es>
"Julián Marcelo" <julian.marcelo@sema.es>
"Julio Berrocal" <berrocal@dit.upm.es>
"Liaquat Khan" <liaquat.khan@gta.multicert.org>
"Luis Fernandez" <codasic@jet.es>
"Luis Jiménez" <infosec@areatec.com>
"Manuel Jacinto Martínez Álvarez" <mjma@notes.banesto.es>
"Marcos López Chávarri" <mlopez@bancopopular.es>
"María José Caro" <carobm@inta.es>
"Miguel Bañón" <bagnonm@safelayer.com>
"María del Mar Solís" <mar.solis@safelayer.com>
"Pedro Pablo López" <pedro_pablo_lopez_rsi@cajarural.com>
"Pino Caballero" <pcaballe@ull.es>
"Rafael Molina Palomo" <rmp1@bancosantander.es>
"Ramon Miralles López" <rmiralles@ctele.gencat.es>
"Roberto Moya Quiles" <rmoya@dimasoft.es>
"Rosa María García Ontoso" <rgo608@comadrid.es>
"Stefan Kelm" <kelm@secorvo.de>
"Teresa Nuñez" <tnunez@europamc.com>
"Tomás Sánchez" <tsancheza@fnmt.es>
"Xavier Sanz" <xsanz@ctele.gencat.es>
```

Contents

List of Figures

List of Tables

Bibliography

[Bor44] J.L. Borges. Ficciones, La biblioteca de Babel. 1944.

[DW97] Georgia DARPA Workshop, Savannah, editor. *Update: CERT/CC Vulnerability Knowledgebase*, 1997.

[fip] *FIPS 140-2 SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES*.

[How97] J. D. Howard. *An Analysis of Security Incidents on the Internet:1989-1995*. PhD thesis, Carnegie Mellon University, 1997.

[Krs98] I.V. Krsul. *Software Vulnerability Analysis*. PhD thesis, Purdue University, 1998.

[Nat00] National Institute of Standards and Technology. *A Proposed Standard for Role-Based Access Control*, December 2000.

[NSA00] NSA, The Mitre Corporation, Mitretek Systems and Sparta, Inc. *The Common Criteria Profiling Knowledge Base*, 1.0i edition, March 2000.

[ottNCSC89] Proceedings of the 12th National Computer Security Conference, editor. *A Survey of Computer Abuse Techniques*, 1989.

[Pow96] R. Power. Current and future danger: A csi primer of computer crime and information warfare. *CSI Bulletin*, 1996.

[PW84] T. Perry and P. Wallich. Can computer crime be stopped? *IEEE Spectrum*, 5(21), 1984.

[the99a] the Common Criteria Project Sponsoring Organisations. *Common Criteria for Information Technology Security Evaluation*, 2.1 final edition, August 1999.

[the99b] the Common Criteria Project Sponsoring Organisations. *Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model*, 2.1 final edition, August 1999.

[the99c] the Common Criteria Project Sponsoring Organisations. *Common Criteria for Information Technology Security Evaluation Part 2: Security functional requirements*, 2.1 final edition, August 1999.

[the99d] the Common Criteria Project Sponsoring Organisations. *Common Criteria for Information Technology Security Evaluation Part 3: Security assurance requirements*, 2.1 final edition, August 1999.

Revision History

Detailed changes between versions of this document can be analyzed by consulting the RCS file,

`http://www.safelayer.com/download/pkipp/9.PPDocuments/PSKPP.tex,v`

```
PSKPP.tex,v
Revision 1.1  2002/04/04 16:44:34  bagnonm
Removed orphan security objective Robust_Encryption
Corrected iff formula
Minor editorial english corrections
```

Acronyms and prefixes

The following prefixes are used to mark concept categories:

AT	Assumption regarding threats
AP	Assumption regarding security policy statements
T	Threat
DA	Detailed Attack
GP	General Security Policy Statement
DP	Detailed Security Policy Statement
O	Security Objective

The following acronyms are used in the document text:

CSP	Critical security parameter
EDC	Error Detection Code
IP	Internet Protocol
PP	Protection Profile
RNG	Random Number Generator
ST	Security Target
TOE	Target of Evaluation
TSC	TSF Scope of Control
TSF	TOE Security Functions
PKI	Public Key Infrastructure
PSK	PKI Secure Kernel

Chapter 1

INTRODUCTION

1.1 Identification

Title "The PKI Secure Kernel Protection Profile"

Reference PSK PP, Version "1.1 evaluated"

Author The "*PKI PP working group*"

1.2 Overview

A number of approaches have been taken to provide PP coverage to the PKI concept and architecture. This PP is the core specification for a family of PPs that address a complex and distributed PKI with advanced security needs.

This document is not concerned with elements of a PKI but with an agreed rationale of how the fundamental PKI assets should be securely handled, in particular the Public Key elements and their operation. This is defined for the rest of the document as the "PKI Secure Kernel".

Note that this PP tries not to include any semantic or purpose to the usual Public Key functionality, since that is intended to be defined by the PKI Application Specific PPs that form the rest of this PP family.

It is expected that new architectural elements will be defined, including uses of Public Key concepts that are not yet known.

A PP for these new concepts can be easily produced by claiming compliance of this PP for the core security services and focusing the architectural element PP in its application-level specifics.

It is expected that STs that claim compliance with this PP family claim compliance, as a minimum, with this PSK PP, and then with as many application-specific PPs as it may implement.

An effort has been made to cover a wide spectrum of threats and policies in the TOE environment. As a result, the derived security requirements form a very secure baseline, possibly applicable to most complex scenarios.

The environmental analysis and derived threat analysis and general policy statements have been done within the domain of knowledge defined in [NSA00], that also guides the derivation of allocated detailed attacks, detailed policy statements and security objectives and functional and assurance security requirements. The original [NSA00] has been refined to correct some minor details, and a supporting tool has been used to help in the construction of this document.

This Protection Profile fully complies with [the99a], being both [the99c] and [the99d] compliant. It also collects fundamental requirements from [fip], that have been used to instantiate general requirements whose applicability has been mandated by [NSA00], aiming to comply with FIPS 140-2 level 4.

No hardware security requirements are incorporated into this PP. Implementers shall decide, on designing their TOEs, the actual hardware protection that they require. At least, they must comply with the stated assumptions, most of them related to hardware issues.

Chapter 2

TOE DESCRIPTION

The TOE, the so called “PKI secure Kernel” is centered around application or purpose independent uses of asymmetric cryptography. An initial set of functions is allocated to the PSK:

1. Key generation.
2. Key filing.
3. Key export.
4. Key import, including certificates.
5. Asymmetric cryptography functionality:
    (a) Encryption and verification
    (b) Decryption and signing

Since this PP is about a protecting layer over this basic usage of cryptographic , the TOE is completely defined by including under its description all the emanating security requirements in Chapter 5.

Actual TOEs claiming compliance of this document may have many possible implementation or design solutions. It may be an integral part of a general application that requires asymmetric cryptography, or it may be an advanced token. It could also be a general dynamically loadable library, and any form is fit for compliance as long as all the requirements and assumptions contained in this PP are met.

Chapter 3

TOE SECURITY ENVIRONMENT

The complete picture at Fig.7.1 shows that to counter with threats and to comply with security policies, security objectives must be met by the TOE.

In some cases, the TOE is able to meet the whole set of security objectives that are required to fully mitigate a complete set of detailed attacks, also known as a threat. In other cases, the TOE, with the set of security functional requirements mandated by this Protection Profile, fails in fulfilling a security objective, and in turn, its associated detailed attacks or security policies.

This chapter about the TOE Security Environment details those security objectives that must be fulfilled by the operational environment, either by procedural or organizational measures, or by additional technical security features.

How this should be done is outside the scope of this document, but it must be clear the set of security objectives not covered by the TOE and the consequences if these are not fulfilled by other means.

By meeting these security objectives, those detailed attacks or security policies not covered by the TOE are fulfilled, and the associated risk can be considered to be mitigated.

Some threats and general security policies are covered by a mixture of TOE and environment responsibilities. This is the case when the TOE is able to counter some of the detailed attacks that define a threat, but not all of them. In this case, the threat or policy is included in this chapter, but only with respect to those unfulfilled security objectives by the TOE.

3.1 Assumptions regarding threats

The following assumptions are considered about the TOE environment and its capability to mitigate the following threats.

3.1.1 Assumptions about the administrative users of the TOE

AT.Admin The following threats are countered by the TOE environment:

T.Admin_UserPriv Administrator violates user privacy policy (See 3.5.1).

In particular, the following attacks are considered:

DA.Admin_UserPriv_Agg Administrator aggregates privacy information, (See 3.6.1). with respect to the following security objective(s):

O.Limit_ObserveRoles Limit observation of service usage to authorized users (See 4.2.5).

O.Prevent_Link Prevent linking of multiple service use (See 4.2.14).

O.Prevent_Observe Prevent observation of service use (See 4.2.15).

DA.Admin_UserPriv_Col Administrator reads collected user privacy information, (See 3.6.2). with respect to the following security objective(s):

O.Prevent_AskPrivInfo Prevent system from collecting user privacy information (See 4.2.13).

DA.Admin_UserPriv_Gen Administrator reads system generated privacy information, (See 3.6.3). with respect to the following security objective(s):

O.Permit_Aliases Permit users to use services under aliases (See 4.2.11).

O.Permit_Anonymity Permit users to use services anonymously (See 4.2.12).

T.Malicious_Code Malicious code exploitation (See 3.5.2).

In particular, the following attacks are considered:

DA.Mal_Code_Hack_Downld Malicious code perpetrator dissemination, (See 3.6.4). with respect to the following security objective(s):

O.Clean_Obj_Recovery Object and data recovery free from malicious code (See 4.2.2).

DA.Mal_Code_Hack_Exe Malicious code perpetrator execution, (See 3.6.5). with respect to the following security objective(s):

O.Clean_Obj_Recovery Object and data recovery free from malicious code (See 4.2.2).

DA.Mal_Code_IT_Download Malicious code accidental IT download, (See 3.6.6). with respect to the following security objective(s):

O.Clean_Obj_Recovery Object and data recovery free from malicious code (See 4.2.2).

DA.Mal_Code_IT_Exe Malicious code IT execution, (See 3.6.7). with respect to the following security objective(s):

O.Clean_Obj_Recovery Object and data recovery free from malicious code (See 4.2.2).

DA.Mal_Code_Usr_Downld Malicious code accidental user download, (See 3.6.8). with respect to the following security objective(s):

O.Clean_Obj_Recovery Object and data recovery free from malicious code (See 4.2.2).

DA.Mal_Code_Usr_Exe Malicious code user execution, (See 3.6.9). with respect to the following security objective(s):

O.Clean_Obj_Recovery Object and data recovery free from malicious code (See 4.2.2).

T.Spoofing Legitimate system services are spoofed (See 3.5.3).

In particular, the following attacks are considered:

DA.Hack_Spoof_Login Login program replicated to capture authentication data, (See 3.6.10). with respect to the following security objective(s):

O.User_Auth_Enhanced Enhanced user authentication (See 4.2.21).

3.1.2 Assumptions about the expected hacking threats over the TOE

AT.Hacker The following threats are countered by the TOE environment:

T.Hack_Avl_Resource Hacker attempts resource denial of service (See 3.5.4).
In particular, the following attacks are considered:

DA.Hack_Prcsr_Overload Hacker causes system task overload resulting in denial of service, (See 3.6.11). with respect to the following security objective(s):

O.React_Discovered_Atk React to discovered attacks (See 4.2.16).

T.Hack_Comm_Eavesdrop Hacker eavesdrops on user data communications (See 3.5.5).
In particular, the following attacks are considered:

DA.Hack_CommEaves_Tap An outsider taps a communications line, (See 3.6.12). with respect to the following security objective(s):

O.Comm_Line_Protection Physical protection of the communications line (See 4.2.3).

O.Tamper_ID Tamper detection (See 4.2.19).

O.Tamper_Resistance Tamper resistance (See 4.2.20).

T.Hack_Masq Hacker masquerading as a legitimate user or as system process (See 3.5.6).
In particular, the following attacks are considered:

DA.Hack_Masq_Wauth Masquerading due to weak authentication, (See 3.6.13). with respect to the following security objective(s):

O.User_Auth_Enhanced Enhanced user authentication (See 4.2.21).

O.User_Auth_Multiple Require multiple authentication mechanisms (See 4.2.22).

T.Hack_Phys Exploitation of vulnerabilities in the physical environment of the system (See 3.5.7).
In particular, the following attacks are considered:

DA.Hack_Phys_Crypto Physical attack on cryptographic assets, (See 3.6.14). with respect to the following security objective(s):

O.Tamper_ID Tamper detection (See 4.2.19).

O.Tamper_Resistance Tamper resistance (See 4.2.20).

DA.Hack_Phys_Damage Hacker physically attacks the system, (See 3.6.15). with respect to the following security objective(s):

O.Tamper_ID Tamper detection (See 4.2.19).

O.Tamper_Resistance Tamper resistance (See 4.2.20).

T.Hack_Social_Engineer Social engineering (See 3.5.8).
In particular, the following attacks are considered:

DA.Hack_SocEng_Password Social engineering to steal password, (See 3.6.16). with respect to the following security objective(s):

O.User_Auth_Enhanced Enhanced user authentication (See 4.2.21).

DA.Hack_SocEng_SysInfo Hacker uses social engineering to learn system information, (See 3.6.17). with respect to the following security objective(s):

O.Audit_Unusual_User Audit unusual user activity (See 4.2.1).

3.1.3 Assumptions about the TOE physical environment

AT.Physical_Environment The following threat is countered by the TOE environment:

T.Component_Failure A critical system component fails (See 3.5.9).

In particular, the following attacks are considered:

DA.Hardware_Flaw System hardware fails during system operation, (See 3.6.18). with respect to the following security objective(s):

O.Fault_Tolerance Provide fault tolerant operations for critical components (See 4.2.4).

DA.Software_Flaw System use uncovers an intrinsic software flaw in a critical system component, (See 3.6.19). with respect to the following security objective(s):

O.Fault_Tolerance Provide fault tolerant operations for critical components (See 4.2.4).

3.1.4 Assumptions about the TOE hardware and software

AT.System_HW_SW The following threat is countered by the TOE environment:

T.Failure_DS_Comp Failure of a distributed system component (See 3.5.10).

In particular, the following attacks are considered:

DA.Failure_DS_Comm Communications function failure, (See 3.6.20). with respect to the following security objective(s):

O.Fault_Tolerance Provide fault tolerant operations for critical components (See 4.2.4).

3.1.5 Assumptions about the TOE user behaviour

AT.User The following threats are countered by the TOE environment:

T.Repudiate_Receive Recipient denies receiving information (See 3.5.11).

In particular, the following attacks are considered:

DA.Repudiate_Rcvr_Int Denial of having received data from another local user, (See 3.6.21). with respect to the following security objective(s):

O.NonRepud_Locals_Rcvd Non-repudiation for received information, local users (See 4.2.9).

DA.Repudiate_Rcvr_Local Denial of having received information from a remote user, (See 3.6.22). with respect to the following security objective(s):

O.NonRepud_Assess_Recd Non-repudiation support for received information by a nonlocal sender's TSF (See 4.2.7).

O.NonRepud_Gen_Recd Non-repudiation support for received information by the recipient's TSF (See 4.2.8).

DA.Repudiate_Rcvr_Rem Denial of having received information by a remote user, (See 3.6.23). with respect to the following security objective(s):

O.NonRepud_Assess_Recd Non-repudiation support for received information by a nonlocal sender's TSF (See 4.2.7).

O.NonRepud_Gen_Recd Non-repudiation support for received information by the recipient's TSF (See 4.2.8).

T.Repudiate_Transact A participant denies performing a transaction (See 3.5.12).

In particular, the following attacks are considered:

DA.Repudiate_Trans_Loc Circumvent non-repudiation in a transaction involving a user and a local system, (See 3.6.24). with respect to the following security objective(s):

O.NonRepud_Locals_Rcvd Non-repudiation for received information, local users (See 4.2.9).

DA.Repudiate_Trans_Uloc Circumvent non-repudiation in a transaction involving a local user and a remote system, (See 3.6.25). with respect to the following security objective(s):

O.NonRepud_Assess_Recd Non-repudiation support for received information by a nonlocal sender's TSF (See 4.2.7).

O.NonRepud_Gen_Recd Non-repudiation support for received information by the recipient's TSF (See 4.2.8).

DA.Repudiate_Trans_Urem Circumvent non-repudiation in a transaction involving a remote user and a local system, (See 3.6.26). with respect to the following security objective(s):

O.NonRepud_Assess_Recd Non-repudiation support for received information by a nonlocal sender's TSF (See 4.2.7).

O.NonRepud_Gen_Recd Non-repudiation support for received information by the recipient's TSF (See 4.2.8).

T.User_Err_Inaccess User error makes data inaccessible (See 3.5.13).

In particular, the following attacks are considered:

DA.User_Err_Delete User error deletes data, (See 3.6.27). with respect to the following security objective(s):

O.Rollback Rollback (See 4.2.17).

T.User_Misuse_Avl_Resc User's misuse causes denial of service (See 3.5.14).

In particular, the following attacks are considered:

DA.User_Obst_Res_Use User obstructs legitimate use of resources., (See 3.6.28). with respect to the following security objective(s):

O.Tamper_ID Tamper detection (See 4.2.19).

O.Tamper_Resistance Tamper resistance (See 4.2.20).

3.2 Assumptions regarding policies

The following assumptions are considered about the TOE environment and its capability to comply with the Detailed Policy Statements indicated.

3.2.1 Assumptions about TOE availability

AP.Availability The following Policies are countered by the TOE environment:

DP.Malicious_Code Malicious code prevention (See 3.10.1).

In particular, the following objectives are considered:

O.Malicious_Code Procedures for preventing malicious code (See 4.2.6).

DP.Sys_Backup_Verify Backup protection and restoration (See 3.10.2).

In particular, the following objectives are considered:

O.Sys_Backup_Verify Detect modifications of backup hardware, firmware, software (See 4.2.18).

3.2.2 Assumptions about TOE access control policies

AP.Information_AC The following Policies are countered by the TOE environment:

DP.User_Auth_Enhanced Enhanced user identification and authentication (See 3.10.3).
In particular, the following objectives are considered:

O.User_Auth_Enhanced Enhanced user authentication (See 4.2.21).

3.2.3 Assumptions about TOE integrity policy

AP.Integrity The following Policies are countered by the TOE environment:

DP.Non-Repudiation Non-repudiation capabilities (See 3.10.4).
In particular, the following objectives are considered:

O.NonRepudiate_Recd Non-repudiation for received information (See 4.2.10).

3.2.4 Assumptions about physical control policies

AP.Physical_Control The following Policies are countered by the TOE environment:

DP.Tamper_ID Physical tampering detection and notification (See 3.10.5).
In particular, the following objectives are considered:

O.Tamper_ID Tamper detection (See 4.2.19).

3.3 Threats addressed by the TOE

The TOE shall be able to counter the general threats that have been identified as applicable and documented herein.

Besides each threat description, most threats and attacks are defined in terms of **Threat Agent, Method and Results**, where applicable or feasible. For example, the Result of a particular Threat may vary as a function of the selected attack, and thus is not defined at Threat level, but referred to the applicable attacks.

Each of these terms is again defined with the following attributes [1];

Agents Threat Agents are characterized in terms of Agent Types, Authentication, Attitude, Motive, Sophistication, Localities, and Forces. The chosen classification approach was influenced by previous classification efforts due to [DW97] (access required), [PW84] (perpetrators), and [How97] (attackers).

Agent Types Most threats are carried out, directly or indirectly, by Human threat agents, but Other agents are possible as well, namely forces of nature. The value Any is included as an option, meaning both Human and Other agents. Most of the other Agent Attributes apply only to human agents.

Authentication Four possibilities are considered: None, Identified, Authenticated, and Privileged. The Privileged value is relevant to administrators who acquire extra privileges via the authentication mechanism.

Attitude, Motive, Sophistication These traits assess strength of attack by a Human agent. The knowledge base entertains the possibility that some systems are so insecure as to contain vulnerabilities that are exploited even by Well Behaved, Constructive users who bring Zero sophistication to bear on the attacks they mount. Other highly secure systems are designed to withstand Deliberate attacks by Hostile threat agents with High sophistication.

[1] Quoted from [NSA00]

Localities Some threat agents are physically Local to the system being attacked, while others are Remote and operate through network connections or other electronic interfaces. This Attribute makes sense for Any agent Type.

Forces Non-human, "Other" agents may be characterized variously as Communications, Disaster, Emanations, Power, Unusual Conditions, or Any.

Methods Threat methods are characterized in terms of Life cycle Phases, Human Roles, Actions performed, and Vulnerabilities and Exposures exploited.

Life cycle Phases Attacks may be mounted at Any Time, i.e., during Development, during Operation, or upon Disposal.

Human Role Only users who have been assigned System Duties can mount some attacks. Others are available to any Service User. Still others are available even to those who have been assigned No Duties at all.

Action This Attribute is unconstrained. Actions performed during an attack tend to be closely aligned with expected results. For Attacks on Security Protection Mechanisms, actions may involve masquerade, subversion, corruption, or trespass. Attacks on Information Integrity may involve Falsification and Error Propagation. Attacks against confidentiality may involve exposure (deliberate disclosure, scavenging), aggregation or inference (cryptographic browsing, query analysis, traffic analysis), interception (eavesdropping, wiretapping, emanations analysis, simple theft), or intrusion (cryptanalysis, reverse engineering, covert channel usage, trespass). Attacks on the Availability of information or services may involve incapacitation, obstruction (interference, exploitation, resource exhaustion), or theft of service. Actions may also be characterized by degree of interference with the attacked system: physical attack, logical intervention, or just observation. For additional approaches to characterizing threat actions, see [Pow96], [Krs98].

Vulnerabilities This Attribute is also unconstrained. It is intended to be fairly broad, including vulnerabilities in both the TOE and its environment, including human vulnerabilities. Typical IT vulnerabilities might involve one or more of the following: Input Validation Error (Boundary Condition Error, Buffer Overflow), Access Validation Error, Exceptional-Condition Handling Error, Environmental Error, Configuration Error, Race Condition. Vulnerabilities may also be described in terms of their location. Approaches to classification of the location of IT vulnerabilities may be found in Krsul's categories, aspects of the scheme in Section 6.1 of Krsul's thesis [Krs98], and the first access scheme in [How97]. Previous efforts to classify human vulnerabilities may be found in [ottNCSC89]. Note that the CC includes functional requirements specifically designed to compensate for human vulnerabilities (e.g., FAU: Security audit, FCO: Communication, FMT: Security management, much of FTA: TOE access). In addition, much of the CC's Part 3 requirements are intended to counter human vulnerabilities in a system's pre-deployment environment.

Results Threat results are characterized in terms of Loss Types, affected IT Capabilities, loss Locations, and affected Security Functions.

Loss Types Loss may involve the loss of Availability, Confidentiality, Integrity, and/or Security Protection. For this characterization of loss types to be fully applicable, it is necessary that these traditional security-classification terms be interpreted quite broadly - see the Glossary for details. These traditional loss categories are attributed to Don Parker of SRI and are described in [Pow96]. Note that some variants of this scheme attempt to omit security-protection threats.

IT Capabilities An attack may involve System capabilities (including security-relevant code or data). Alternatively it may involve User Data and/or User Processing.

Locations Losses resulting from an attack may occur in the system itself (i.e., the TOE) and/or in the system's Environment. The latter value is particularly relevant in the case of network firewalls.

Security Functions If an attack involves loss of security protection, it is relevant to ask what types of security protection are affected. We have used CC Classes as a convenient way to classify the types of security protection that might be lost.

3.3.1 Administrative errors of commission

T.Admin_Err_Commit An administrator commits errors that directly compromise organizational security objectives or change the technical security policy enforced by the system or application.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**
Human	Privileged	Accidental	Constructive	Low

Method

Lifecycle Phases	**Human Role**	**Action**
Operation	System Duties	Error

3.3.2 Administrative errors of omission

T.Admin_Err_Omit The system administrator fails to perform some function essential to security.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**
Human	Privileged	Accidental	Constructive	Low

Method

Lifecycle Phases	**Human Role**
Operation	System Duties

3.3.3 Hostile administrator modification of user or system data

T.Admin_Hostile_Modify An administrator maliciously obstructs organizational security objectives or modifies the system's configuration to allow security violations to occur.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**
Human	Privileged	Deliberate	Hostile	Low

Method

Lifecycle Phases	**Human Role**	**Action**
Operation	System Duties	Modify or destroy TSF code or data

3.3.4 Software containing security-related flaws

T.Dev_Flawed_Code A system or applications developer delivers code that does not perform according to specifications or contains security flaws.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	Privileged	Accidental	Constructive	Low

Method

Lifecycle Phases	Human Role
Development	System Duties

3.3.5 Hacker undetected system access

T.Hack_AC A hacker gains undetected access to a system due to missing, weak and/or incorrectly implemented access control causing potential violations of integrity, confidentiality, or availability.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	Authenticated	Deliberate	Hostile	Moderate

Method

Lifecycle Phases	Human Role
Operation	Service User

3.3.6 Message content modification

T.Hack_Msg_Data A hacker modifies information intercepted from a communication link between two unsuspecting entities before passing it on, thereby deceiving the intended recipient.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	Moderate	Remote

Method

Lifecycle Phases	Human Role
Operation	No Duties

Result

Loss Types	IT Capabilities
Integrity	User Data

3.3.7 Unexpected disruption of system or component power

T.Power_Disrupt A human or environmental agent disrupts power causing the system to lose information or security protection.

Agent

Agent Type	Forces
Other	Power

Method

Lifecycle Phases
Operation

Result

Loss Types	IT Capabilities
Availability	System

3.3.8 Sender denies sending information

T.Repudiate_Send The sender of a message denies sending the message to avoid accountability for sending the message or to avoid obligations incurred as a result of sending the message.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	Authenticated	Deliberate	Negligent	Low

Method

Lifecycle Phases	Human Role
Operation	Service User

Result

Loss Types	IT Capabilities
Integrity	User Data

3.3.9 Hostile user acts cause confidentiality breaches

T.User_Abuse_Conf A user collects sensitive or proprietary information and removes it from the system.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	Authenticated	Deliberate	Hostile	Low

Method

Lifecycle Phases	Human Role
Operation	Service User

Result

Loss Types	IT Capabilities
Confidentiality	User Data

3.3.10 User abuses authorization to collect data

T.User_Collect User abuses granted authorizations to improperly collect sensitive or security-critical data.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	Authenticated	Deliberate	Hostile	Low

Method

Lifecycle Phases	Human Role	Action
Operation	Service User	Observe

Result

Loss Types	IT Capabilities
Confidentiality	User Data

3.3.11 User errors cause confidentiality breaches

T.User_Err_Conf A user commits errors that cause information to be delivered to the wrong place or wrong person.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	Authenticated	Accidental	Constructive	Zero

Method

Lifecycle Phases	Human Role
Operation	Service User

Result

Loss Types	IT Capabilities
Confidentiality	User Data

3.3.12 User errors cause integrity breaches

T.User_Err_Integrity A user commits errors that induce erroneous actions by the system and/or erroneous statements its users.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	Authenticated	Accidental	Constructive	Zero

Method

Lifecycle Phases	Human Role
Operation	Service User

Result

Loss Types	IT Capabili-ties
Integrity	User Data

3.3.13 User errors undermine the system's security features

T.User_Err_Slf_Protect A user commits errors that cause the system or one of its applications to undermine the system's security features.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	Authenticated	Accidental	Constructive	Zero

Method

Lifecycle Phases	Human Role
Operation	Service User

3.3.14 User abuses authorization to modify data

T.User_Modify A user abuses granted authorizations to improperly change or destroy sensitive or security-critical data.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	Authenticated	Accidental	Constructive	Zero

Method

Lifecycle Phases	Human Role
Operation	Service User

Result

Loss Types	IT Capabili-ties
Integrity	User Data

3.3.15 User abuses authorization to send data

T.User_Send A user abuses granted authorizations to improperly send sensitive or security-critical data.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	Authenticated	Accidental	Constructive	Zero

Method

Lifecycle Phases	Human Role
Operation	Service User

Result

Loss Types	IT Capabilities
Confidentiality, Integrity	User Data

3.4 Detailed Attacks countered by the TOE

The TOE shall be able to counter the detailed attacks that have been identified as applicable and documented herein.

3.4.1 Accidental mismanagement of cryptographic functions

DA.Adm_Err_Crypto An administrator misconfigures cryptographic functions or stores plaintext keys in insecure areas.

3.4.2 Destruction or modification of audit data

DA.Adm_Hstl_Audit_Dstr An administrator seeks to cover up misbehavior by destroying and/or falsifying audit data.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Deliberate	Hostile	Moderate	Any

Method

Lifecycle Phases	Human Role	Action	Vulnerabilities
Operation	System Duties	Destroy or falsify	Inadequate protection of audit data

Result

Loss Types	IT Capabilities	Locations	Security Functions
Security Protection	System	TOE	FAU

3.4.3 Administrator modifies or destroys user data or applications

DA.Adm_Hstl_Mod_DataAps The administrator abuses IT or user trust, as being the administrator and without changing the user imposed data security attributes, by destroying data or applications for malicious reasons or to cover up misappropriate behavior.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Deliberate	Hostile	Low	Any

Method

Lifecycle Phases	Human Role	Action
Operation	System Duties	Modification of user data or applications

Result

Loss Types	IT Capabilities	Locations
Availability, Integrity	User Data	TOE

3.4.4 Administrator maliciously modifies or deletes data access control attributes

DA.Adm_Hstl_Mod_Data_AC An administrator maliciously modifies access control attributes, allowing the administrator or other perpetrator to gain access and manipulative capability to organizational assets, contrary to organizational policy.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Deliberate	Hostile	Low	Any

Method

Lifecycle Phases	Human Role	Action
Operation	System Duties	Modify access control attributes

Result

Loss Types	IT Capabilities	Locations	Security Functions
Security Protection	System, User Data	TOE	FDP

3.4.5 The administrator maliciously modifies information flow control.

DA.Adm_Hstl_Mod_IFC The administrator maliciously alters information flow control policy to allow information to flow to inappropriate locations for unauthorized users access or modification.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Deliberate	Hostile	Moderate	Any

Method

Lifecycle Phases	Human Role	Action
Operation	System Duties	Modification of Information Flow Control

Result

Loss Types
Any

3.4.6 Administrator maliciously modifies system entry parameters

DA.Adm_Hstl_Mod_SEP An administrator or user masquerading as an administrator maliciously modifies system entry parameters which would allow unauthorized access to an organization's protected assets.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Deliberate	Hostile	Low	Any

Method

Lifecycle Phases	Human Role	Action
Operation	System Duties	Modification of system entry parameters

Result

Loss Types
Security Protection

3.4.7 Administrator maliciously modifies security-critical code

DA.Adm_Hstl_Mod_TSFCode The administrator modifies the security-critical (TSF) code to weaken the security effectiveness of the TSF or introduce a new security breech.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Deliberate	Hostile	Moderate	Any

Method

Lifecycle Phases	Human Role	Action
Operation	System Duties	Modify TSF or system code

Result

Loss Types	IT Capabilities	Security Functions
Security Protection	System	FPT

3.4.8 Administrator maliciously modifies user/subject bindings

DA.Adm_Hstl_Mod_USB The administrator modifies a user/subject binding which would allow a user to act on an object without creating an audit trail.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Deliberate	Hostile	High	Any

Method

Lifecycle Phases	Human Role	Action
Operation	System Duties	Modification of user/subject binding

Result

Loss Types
Security Protection

3.4.9 Administrator maliciously modifies user attributes and/or roles

DA.Adm_Hstl_Mod_UsrAttr The administrator modifies or mishandles the users attributes or roles which allows users, unauthorized or authorized, to have the ability to perform inappropriate actions or could prevent a user from performing an authorized action.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Deliberate	Hostile	Low	Any

Method

Lifecycle Phases	Human Role	Action
Operation	System Duties	Modification of user attributes and/or roles

Result

Loss Types
Any

3.4.10 User privileges and/or authorizations are not updated upon reassignment

DA.Adm_Misconfig_User A change in the status of users duties do not get reflected in administratively controlled privileges and/or authorizations.

3.4.11 Administrator error modifies access control or information flow policy

DA.Admin_Err_AC_Policy An administrator's error in data entry changes the access control or information flow policy enforced by the system in such a way that it no longer serves its intended purpose.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Accidental	Negligent	Zero	Any

Result

Loss Types	IT Capabilities	Locations	Security Functions
Security Protection	System	TOE	FDP

3.4.12 Administrator error changes audit behavior

DA.Admin_Err_Audit An administrator's error in data entry changes the audit behavior of the system in such a way that auditing no longer serves its intended purpose.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Accidental	Constructive	Zero	Any

Method

Lifecycle Phases	Human Role	Action
Operation	System Duties	Error

Result

Loss Types	Security Functions
Security Protection	FAU

3.4.13 Administrator error modifies authentication enforcement

DA.Admin_Err_Authentic An administrator's error in data entry changes the authentication-enforcement mechanism of the system in such a way that it no longer serves its intended purpose.

Agent

Agent Type	Authentication	Attitude	Motive	Localities
Human	Privileged	Accidental	Constructive	Any

Method

Lifecycle Phases	Human Role	Action
Operation	System Duties	Error

Result

Loss Types
Security Protection

3.4.14 Administrator error makes information unavailable

DA.Admin_Err_Info An administrator's error in data entry makes system or application information unavailable.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Accidental	Constructive	Zero	Any

Method

Lifecycle Phases	Human Role	Action
Operation	System Duties	Error

Result

Loss Types
Availability

3.4.15 Back door left open

DA.Admin_Err_Omit_Trap An administrator inadvertently leaves a back door port open after routine maintenance, allowing continuing unauthorized access by the service organization.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Accidental	Constructive	Zero	Any

Method

Lifecycle Phases	Human Role	Action	Vulnerabilities
Operation	System Duties	Inaction	administrator frailty

Result

Loss Types	Locations
Security Protection	TOE

3.4.16 Administrator error makes resource unavailable

DA.Admin_Err_Resource An administrator's error in data entry makes system or application resources unavailable.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**	**Localities**
Human	Privileged	Accidental	Constructive	Zero	Any

Method

Lifecycle Phases	**Human Role**	**Action**
Operation	System Duties	Error

Result

Loss Types
Availability

3.4.17 Administrator error modifies entry policy

DA.Admin_Err_Sys_Entry An administrator's error in data entry changes the intended entry policy of the system or application.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**	**Localities**
Human	Privileged	Accidental	Constructive	Zero	Any

Method

Lifecycle Phases	**Human Role**	**Action**
Operation	System Duties	Error

Result

Loss Types
Security Protection

3.4.18 Administrator fails to update security configuration

DA.Admin_Err_Update The organizational security policies changes but these changes are not reflected in all system configurations, resulting in circumvention and/or incorrect application of security policies.

3.4.19 Administrator error modifies user security attributes

DA.Admin_Err_User_Attr An administrator's error in data entry modifies a user's security attributes, which makes the attributes inappropriate under the security policy of the system or application.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**	**Localities**
Human	Privileged	Accidental	Constructive	Zero	Any

Method

Lifecycle Phases	**Human Role**	**Action**
Operation	System Duties	Error

Result

Loss Types
Availability, Security Protection

3.4.20 Inconsistent interpretation of audit data attributes

DA.Dev_FC_Attr_Interp The security-critical (TSF) components inconsistently interpret audit data attributes exchanged with another trusted IT product.

3.4.21 Buffers not cleared by the system

DA.Dev_FC_Buff_Not_Clr The system leaves user information in a system buffer for view by another unauthorized user.

3.4.22 Incorrect modification of control data

DA.Dev_FC_Ctrl_Data A security-critical (TSF) component incorrectly modifies control data regarding a user process.

3.4.23 System data incorrectly exchanged

DA.Dev_FC_Data_Export The system incorrectly exchanges system data with another trusted system.

3.4.24 Non-secure recovery

DA.Dev_FC_Recovery A system failure may alter the behavior of the system's security functions in such a way that, upon recovery, it no longer properly enforces its security policy (TSP).

3.4.25 Inaccurate system-data replication

DA.Dev_FC_Replication The system does not accurately replicate system data to different parts of the system where replication is required.

3.4.26 System modification by unauthorized source

DA.Dev_FC_Self_Protect Software developer or hacker modifies system security functions resulting in a loss of security protection.

3.4.27 Malicious developer creates secret trapdoor in system

DA.Dev_FC_Trap_Door The system developer creates a secret back door in the system (TOE) that allows covert access by the developer. This allows the developer to collect information, monitor user actions, modify the operation of the TOE, or just make unauthorized use of the TOE.

Agent

Agent Type	Authentication	Attitude	Motive	Localities
Human	Identified	Deliberate	Negligent	Any

Method

Action
install trap door

3.4.28 Hacker gains access through a vulnerability in code

DA.Hack_AC_Code_Vul The hacker can use vulnerabilities found in system or application code to break into a system undetected.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	Moderate	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Maliciously circumvents access control through a code vulnerability

3.4.29 Weak system access control mechanism or system access control implementation

DA.Hack_AC_Weak The system access control mechanism(s) or user attributes are weak and can be broken or the implementation methods of the system access control causes the weakness.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Deliberate	Hostile	Moderate	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Penetrate weak access control

Result

Loss Types
Any

3.4.30 The communication mechanism emanates data

DA.Hack_CommEaves_Eman An outsider uses special equipment to capture emanations off the communications line.

3.4.31 Outsider intercepts user communications

DA.Hack_CommEaves_Intrc An outsider who is not an intended recipient intercepts user data communications.

3.4.32 Hacker causes overload of communication resources

DA.Hack_Comm_Overload The unauthorized use of communication resources by a hacker causes a denial or delay in service to legitimate operations within the TOE scope of control. This would include the excess bandwidth utilization, leading to the TOE's inability to perform it's security functions.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	Moderate	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Excessively use communication resources

Result

Loss Types
Availability, Security Protection

3.4.33 Accidental or deliberate mishandling of cryptographic assets external to the TOE

DA.Hack_Ext_CryptoAsset Cryptographic assets are mishandled after the leave the TOE, either in transit or while residing on stored media.

3.4.34 A hacker assumes the identity of an authorized user

DA.Hack_Masq_Hijack A hacker captures the interactive session of an authorized user. The hacker now appears as a legitimate user and can perform any action allowed to that user, including reading or modifying sensitive data.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	High	Remote

Method

Action
Penetrate a communications link

Result

Loss Types	Locations
Any	Any

3.4.35 A user assumes the identity of an authorized user

DA.Hack_Masq_Uwkstn An individual takes advantage of an unattended but active workstation to perform operations in the name of the logged-in user. Such operations may include some operations that the attacker is not normally allowed to perform.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Negligent	Low	Local

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Usurp an active session on a workstation

Result

Loss Types
Any

3.4.36 Modification of security-critical data in transit from a remote trusted site

DA.Hack_MsgData_RcvTSF Security-critical (TSF) data is modified in transit from a remote trusted site, either accidentally by the communications infrastructure or deliberately by a hostile outsider.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	High	Remote

Method

Action	Vulnerabilities
Message modification	Inadequate TSF data protection by the remote site and/or inadequate data validation by the TOE

Result

Loss Types	IT Capabilities	Locations
Integrity, Security Protection	System	TOE

3.4.37 Modification of user data in transit from a remote site

DA.Hack_MsgData_RcvUsr A hostile outsider modifies message data in route to the system. Alternatively, errors in the communications infrastructure modify the message.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	High	Remote

Method

Lifecycle Phases	Human Role	Action	Vulnerabilities
Operation	No Duties	Message modification	Inadequate user data protection by the remote site and/or inadequate data validation by the TOE

Result

Loss Types	Locations
Integrity	TOE

3.4.38 Modification of security-critical data in transit to a remote site

DA.Hack_MsgData_SndTSF Security-critical (TSF) data is modified in transit to a remote site, either accidentally by the communications infrastructure or deliberately by a hostile outsider.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	High	Remote

Method

Lifecycle Phases	Human Role	Action	Vulnerabilities
Operation	No Duties	Message modification	Inadequate TSF data protection by the TOE and/or inadequate data validation by the remote site

Result

Loss Types	IT Capabilities	Locations
Integrity, Security Protection	System	Environment

3.4.39 Modification of user data in transit to a remote site

DA.Hack_MsgData_SndUsr A hostile outsider modifies message data in route to a remote site. Alternatively, errors in the communications infrastructure modify the message.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	High	Remote

Method

Lifecycle Phases	Human Role	Action	Vulnerabilities
Operation	No Duties	Message modification	Inadequate user data protection by the TOE and/or inadequate data validation by the remote site

Result

Loss Types	IT Capabilities	Locations
Integrity	User Data	Environment

3.4.40 Attacker modifies protocol headers

DA.Hack_Spoof_MsgHdr An attacker may modify protocol headers such that a user believes the communication is coming from a source that is different from where it was actually sent.

3.4.41 Hacker activities cause storage overload

DA.Hack_Stg_Overload A hacker initiates processes that tax the amount of storage available in the system (TOE). Such would be the case when a hacker floods the TOE with e-mails.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	Moderate	Any

Method

Lifecycle Phases	Human Role
Operation	No Duties

Result

Loss Types
Availability

3.4.42 Resource depletion failure

DA.Phys_CompFail_Res A system allocates so many resources that not enough are left for a critical component to function correctly.

3.4.43 Unexpected power reset

DA.Power_Disrupt_Reset An unintentional, malicious, or environmentally caused power reset occurs, resulting in the loss of critical information or the system to enter a non-secure state.

3.4.44 Denial of having sent information to another local user

DA.Repudiate_Send_Int A local, authorized user sends a message to another local user via the system, and then denies having done it. This affects the recipient of the message as well as any resources allocated or modified by the recipient in response to the message.

3.4.45 Denial of having sent information to a remote user

DA.Repudiate_Send_Local A local, authorized user sends a message to another user at a remote trusted product, and then denies having done it. This affects the recipient of the message as well as any resources allocated or modified by the recipient in response to the message.

3.4.46 Denial of having sent data by a remote user

DA.Repudiate_Send_Rem A local, authorized user receives a message from another user at a remote trusted product who then denies having sent it. This affects the recipient of the message as well as any resources allocated or modified by the recipient in response to the message.

3.4.47 Accidental release of cryptographic assets due to TSF flaw or malfunction

DA.TSF_Err_Conf_Crypto The TSF accidentally releases sensitive plaintext data, red keys, or other cryptographic assets to an inappropriate audience.

3.4.48 User smuggles data using removable media

DA.User_Abuse_Conf_Disk A user collects sensitive or proprietary information and improperly removes it from the system by putting it on removable media.

3.4.49 Steganographic data smuggling

DA.User_Abuse_Conf_Steg An authorized user hides sensitive information in an innocuous-appearing file, for the purpose of covertly passing it to an unauthorized party. The hidden data is undetectable to anyone using the file for its intended purpose, but can be recovered using special techniques.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Accidental	Negligent	Low	Any

Method

Lifecycle Phases	Human Role	Action	Vulnerabilities
Operation	No Duties	Data hiding	Human frailty, inadequate data protection

Result

Loss Types
Confidentiality

3.4.50 User collects data by browsing

DA.User_Collect_Browse An authorized user abuses granted authorizations by browsing files in order to collect data.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**	**Localities**
Human	Authenticated	Deliberate	Hostile	Low	Any

Method

Lifecycle Phases	**Human Role**	**Action**
Operation	No Duties	Observe

Result

Loss Types
Confidentiality

3.4.51 User collects authentication data by deception

DA.User_Collect_Deceive An authorized user steals authentication data by emulating a login procedure on an active terminal.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**	**Localities**
Human	Authenticated	Deliberate	Hostile	Low	Any

Method

Lifecycle Phases	**Human Role**	**Action**
Operation	No Duties	Deception

Result

Loss Types
Confidentiality, Security Protection

3.4.52 User collects data by deduction

DA.User_Collect_Deduce An authorized user abuses granted authorizations by repeatedly accessing aggregate data in order to deduce specific, sensitive data.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**	**Localities**
Human	Authenticated	Deliberate	Hostile	Moderate	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Read repeatedly

Result

Loss Types
Confidentiality

3.4.53 User collects data by eavesdropping

DA.User_Collect_Eaves An authorized user abuses granted authorizations by eavesdropping on communication lines in order to collect data.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Deliberate	Hostile	Low	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Observe

Result

Loss Types
Confidentiality

3.4.54 User collects residual data

DA.User_Collect_Residue An authorized user collects residual data from public objects after prior usage.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Deliberate	Hostile	Moderate	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Observe residual data

Result

Loss Types
Confidentiality

3.4.55 User's unauthorized use causes overload of communication resources

DA.User_Comm_Overload An authorized user exceeds the authorized use of communication resources during the system (TOE) operation. This causes a denial or delay in service to legitimate operations within the TOE scope of control.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Accidental	Negligent	Zero	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Create excessive communication traffic

Result

Loss Types
Availability

3.4.56 Denial of service due to exhausted audit storage

DA.User_ErrAvl_AudExhst An authorized user's actions generate so many audit records that audit storage space is exhausted and the system subsequently denies further service until audit storage becomes available.

3.4.57 Falsification of information quality in data export

DA.User_Err_AttrXpt An authorized user presents incorrect information, indicating to the recipient that it is correct, thereby encouraging the recipient to make unwarranted use of the information.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Accidental	Negligent	Zero	Any

Method

Lifecycle Phases	Human Role	Action	Vulnerabilities
Operation	No Duties	Misrepresentation of security attributes	Human ignorance or inattention, inadequate data labeling

Result

Loss Types
Integrity, Security Protection

3.4.58 Under-classification of data sensitivity on export

DA.User_Err_Conf_Class An authorized user presents confidential or classified information to a recipient, indicating that it is less sensitive than it really is, thereby encouraging the recipient to pass it along to other potentially inappropriate recipients.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**	**Localities**
Human	Authenticated	Accidental	Negligent	Zero	Any

Method

Lifecycle Phases	**Human Role**	**Action**	**Vulnerabilities**
Operation	No Duties	Falsification of security attributes	Human frailty, inadequate data protection

Result

Loss Types
Confidentiality

3.4.59 Accidental release of cryptographic assets due to user error

DA.User_Err_Conf_Crypto User error causes release of cryptographic assets to unauthorized recipients.

3.4.60 Confidentiality violation of export control policy

DA.User_Err_Conf_Exp An authorized user exposes or exports data in violation of export control policy. The data may be private or classified, the recipient is not authorized to receive it.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**	**Localities**
Human	Authenticated	Deliberate	Negligent	Zero	Any

Method

Lifecycle Phases	**Human Role**	**Action**	**Vulnerabilities**
Operation	No Duties	Deliberate export	Human frailty, inadequate data protection

Result

Loss Types
Confidentiality

3.4.61 User error modifying attributes availability

DA.User_Err_Mod_Attr An authorized user erroneously modifies the initial security attributes of user data, which makes the data inaccessible.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Accidental	Constructive	Zero	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Error

Result

Loss Types	Locations
Availability	TOE

3.4.62 Failure to provide object security attributes in data export

DA.User_Err_MsngAttrXpt An authorized user deliberately or accidentally exports data so that the data is not accompanied by required handling information.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Accidental	Constructive	Zero	Any

Method

Lifecycle Phases	Human Role	Action	Vulnerabilities
Operation	No Duties	Withholding of security-attribute information	Human frailty, inadequate data protection

Result

Loss Types
Confidentiality, Integrity

3.4.63 Incorrectly set object attributes

DA.User_Err_Object_Attr An authorized user sets an object's security attributes inappropriately, misdirecting its use. The misdirection may allow unauthorized reading or modification, or it may prohibit authorized reading or modification.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Accidental	Constructive	Zero	Any

Method

Lifecycle Phases	Human Role	Action	Vulnerabilities
Operation	No Duties	Setting object security attributes	Human ignorance or inattention, inadequate data labeling mechanism

Result

Loss Types
Security Protection

3.4.64 User error setting attributes availability

DA.User_Err_Set_Attr An authorized user erroneously sets the initial security attributes of user data, which makes the data inaccessible.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Accidental	Constructive	Zero	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Error

Result

Loss Types
Availability

3.4.65 User modifies audit trail

DA.User_Modify_Audit An authorized user modifies audit data or audit attributes to avoid accountability.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Deliberate	Hostile	Moderate	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Modify TSF data

Result

Loss Types
Integrity, Security Protection

3.4.66 User improperly modifies authentication data

DA.User_Modify_Auth An authorized user changes the authentication data of another user without first masquerading as that user, in a manner that is not consistent with organizational security policy.

Agent

Agent Type	Authentication	Attitude	Motive	Localities
Human	Authenticated	Deliberate	Negligent	Any

Method

Action
Modify TSF data

Result

Loss Types
Integrity, Security Protection

3.4.67 User improperly modifies user data

DA.User_Modify_Data An authorized user modifies or deletes user data in violation of organizational policy.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Deliberate	Negligent	Zero	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Modify

Result

Loss Types	IT Capabilities	Locations
Integrity	User Data	Any

3.4.68 User improperly modifies TSF data

DA.User_Modify_TSFData User modifies or deletes TSF data undermining security protection.

3.4.69 User's unauthorized actions over-task the system causing processor overload

DA.User_Prcsr_Overload The system (TOE) has been over-tasked and can not complete the assigned tasking at all or in an expected amount of time. The user invokes processing functions in association with unauthorized activity that leads to overburdening processing resources on the TOE.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Accidental	Negligent	Zero	Any

Method

Lifecycle Phases	Human Role	Action	Vulnerabilities
Operation	No Duties	Obstruction / Overload	System / Resource Utilization

Result

Loss Types	IT Capabilities	Locations
Availability	System	TOE

3.4.70 User sends data violating confidentiality

DA.User_Send_Conf An authorized user abuses granted authorizations and violates export control policy by sending data to a recipient who is not authorized to receive the data.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Accidental	Negligent	Zero	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Send

Result

Loss Types	IT Capabilities	Locations
Confidentiality	User Data	TOE

3.4.71 User sends data violating integrity

DA.User_Send_Integrity An authorized user deliberately exports data inappropriately, with the result that there is a lack of required quality control on the exported data.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Accidental	Negligent	Zero	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Send

Result

Loss Types
Integrity

3.4.72 User's unauthorized actions cause storage overload

DA.User_Stg_Overload An authorized user's unauthorized use of data storage causes a shortage of disk space for other users.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Accidental	Negligent	Zero	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Prevent Use of Storage by exploiting storage capacity limits

Result

Loss Types
Availability

3.5 Threats addressed by the TOE with support from the environment

The TOE, with the help of the identified Assumptions, shall be able to counter the general threats that have been identified as applicable and documented herein.

3.5.1 Administrator violates user privacy policy

T.Admin_UserPriv An administrator learns the identity (or other privacy related information) of user(s) in violation of user privacy policy. Privacy-related information is sensitive information associated with the identity of a user.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	Privileged	Accidental	Negligent	Low

Method

Lifecycle Phases	Human Role
Operation	System Duties

Result

Loss Types	IT Capabilities
Confidentiality	User Data

3.5.2 Malicious code exploitation

T.Malicious_Code An authorized user, IT system, or hacker downloads and executes malicious code, which causes abnormal processes that violate the integrity, availability, or confidentiality of system assets.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	None	Deliberate	Hostile	Zero

Method

Lifecycle Phases	Human Role
Operation	No Duties

3.5.3 Legitimate system services are spoofed

T.Spoofing An attacker tricks users into interacting with spurious system services.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	Moderate	Remote

Method

Lifecycle Phases	Human Role
Operation	No Duties

3.5.4 Hacker attempts resource denial of service

T.Hack_Avl_Resource A hacker executes commands, sends data, or performs other operations that make system resources unavailable to system users. Resources that may be denied to users include bandwidth, processor time, memory, and data storage.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	None	Deliberate	Hostile	Moderate

Method

Lifecycle Phases	Human Role
Operation	No Duties

Result

Loss Types	IT Capabilities
Availability	System

3.5.5 Hacker eavesdrops on user data communications

T.Hack_Comm_Eavesdrop Hacker obtains user data by eavesdropping on communications lines.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	Low	Remote

Method

Lifecycle Phases	Human Role
Operation	No Duties

Result

Loss Types	IT Capabilities
Confidentiality	User Data

3.5.6 Hacker masquerading as a legitimate user or as system process

T.Hack_Masq A hacker masquerades as an authorized user to perform operations that will be attributed to the authorized user or a system process.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	High	Remote

Method

Lifecycle Phases	Human Role	Vulnerabilities
Operation	No Duties	Inadequate authentication mechanisms and inadequate protection of communications media.

Result

Loss Types	IT Capabilities
Availability, Confidentiality, Integrity	User Data

3.5.7 Exploitation of vulnerabilities in the physical environment of the system

T.Hack_Phys A hacker physically interacts with the system to exploit vulnerabilities in the physical environment, resulting in arbitrary security compromises.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	Moderate	Local

Method

Lifecycle Phases	Human Role
Operation	No Duties

3.5.8 Social engineering

T.Hack_Social_Engineer A hacker uses social engineering techniques to gain information about system entry, system use, system design, or system operation.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	None	Deliberate	Hostile	Zero

Method

Lifecycle Phases	Human Role
Operation	No Duties

3.5.9 A critical system component fails

T.Component_Failure Failure of one or more system components results in the loss of system-critical functionality.

Agent

Agent Type	Forces
Other	Unusual Conditions

Method

Lifecycle Phases
Operation

Result

Loss Types	IT Capabilities
Availability	System

3.5.10 Failure of a distributed system component

T.Failure_DS_Comp Failure of a component that is part of a distributed system will cause other parts of the distributed system to malfunction or provide unreliable results.

Agent

Agent Type	Forces
Other	Unusual Conditions

Method

Lifecycle Phases
Operation

Result

Loss Types	**IT Capabilities**
Availability	System

3.5.11 Recipient denies receiving information

T.Repudiate_Receive The recipient of a message denies receiving the message, to avoid accountability for receiving the message or to avoid obligations incurred as a result of receiving the message.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**
Human	Authenticated	Deliberate	Negligent	Low

Method

Lifecycle Phases	**Human Role**
Operation	Service User

Result

Loss Types	**IT Capabilities**
Integrity	User Data

3.5.12 A participant denies performing a transaction

T.Repudiate_Transact A participant in a transaction denies participation in the transaction to avoid accountability for the transaction or for resulting obligations.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**
Human	Authenticated	Deliberate	Negligent	Low

Method

Lifecycle Phases	**Human Role**
Operation	Service User

Result

Loss Types	**IT Capabilities**
Integrity	User Data

3.5.13 User error makes data inaccessible

T.User_Err_Inaccess A user accidentally deletes user data or changes system data rendering user data inaccessible.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	Authenticated	Accidental	Constructive	Zero

Method

Lifecycle Phases	Human Role	Action
Operation	Service User	Error

Result

Loss Types	IT Capabilities
Availability	User Data

3.5.14 User's misuse causes denial of service

T.User_Misuse_Avl_Resc A user's unauthorized use of resources causes an undue burden on an affected resource.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication
Human	Authenticated	Accidental	Constructive	Zero

Method

Lifecycle Phases	Human Role
Operation	Service User

Result

Loss Types	IT Capabilities
Availability	System

3.6 Detailed Attacks countered by the Environment

The environment, in applications of the stated assumptions, shall be able to counter the detailed attacks that have been identified as applicable and documented herein.

3.6.1 Administrator aggregates privacy information

DA.Admin_UserPriv_Agg An administrator aggregates information that indirectly reveals the identity (or other privacy related information) of user(s) in violation of user privacy policy.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Deliberate	Hostile	Moderate	Any

Method

Lifecycle Phases	Human Role	Action
Operation	System Duties	Aggregate user data

Result

Loss Types	Locations
Confidentiality	Any

3.6.2 Administrator reads collected user privacy information

DA.Admin_UserPriv_Col An administrator reads information collected by the IT system or product that reveals the identity (or other privacy related information) of user(s) in violation of user privacy policy.

3.6.3 Administrator reads system generated privacy information

DA.Admin_UserPriv_Gen An administrator reads information generated by the IT system or product that directly reveals the identity (or other privacy related information) of user(s) in violation of user privacy policy.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Privileged	Deliberate	Hostile	Low	Any

Method

Lifecycle Phases	Human Role	Action
Operation	System Duties	Observe system data

Result

Loss Types	Locations
Confidentiality	TOE

3.6.4 Malicious code perpetrator dissemination

DA.Mal_Code_Hack_Downld A perpetrator disseminates malicious code via push or pull mechanism.

3.6.5 Malicious code perpetrator execution

DA.Mal_Code_Hack_Exe A perpetrator executes malicious code either remotely or locally.

3.6.6 Malicious code accidental IT download

DA.Mal_Code_IT_Download An IT device accidentally transfers or downloads malicious code to itself or other device that it can influence.

3.6.7 Malicious code IT execution

DA.Mal_Code_IT_Exe An IT device under normal operations enters a state required to execute the malicious code.

3.6.8 Malicious code accidental user download

DA.Mal_Code_Usr_Downld An authorized user accidentally downloads malicious code.

3.6.9 Malicious code user execution

DA.Mal_Code_Usr_Exe An authorized user executes malicious code accidentally.

3.6.10 Login program replicated to capture authentication data

DA.Hack_Spoof_Login An attacker simulates the system's login program and runs it at an open terminal or workstation in order to capture a legitimate user's authentication data.

3.6.11 Hacker causes system task overload resulting in denial of service

DA.Hack_Prcsr_Overload Hacker causes system task overload resulting in denial of service. The system (TOE) has been over-tasked and can not complete the assigned tasking at all or in an expected amount of time. The hacker invokes processing functions in association with unauthorized activity that leads to overburdening processing resources on the TOE.

Agent

Agent Type	**Authentication**	**Attitude**	**Motive**	**Sophistication**	**Localities**
Human	Authenticated	Deliberate	Hostile	Low	Any

Method

Lifecycle Phases	**Human Role**	**Action**	**Vulnerabilities**
Operation	No Duties	Deliberate processor overload	System / Resource Utilization

Result

Loss Types
Availability

3.6.12 An outsider taps a communications line

DA.Hack_CommEaves_Tap An outsider uses a device to physically tap the communications line.

3.6.13 Masquerading due to weak authentication

DA.Hack_Masq_Wauth Services are provided to a user application without adequate authentication of the client requesting the service. This would permit someone to receive services for which they are not authorized. However, the server would see them as a legitimate user, which is why this is classified as a masquerade attack.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	None	Deliberate	Hostile	Moderate	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	exploiting weak authentication mechanisms

Result

Loss Types
Any

3.6.14 Physical attack on cryptographic assets

DA.Hack_Phys_Crypto Physical attack causes damage to cryptographic functions and/or release of cryptographic assets

3.6.15 Hacker physically attacks the system

DA.Hack_Phys_Damage Hacker physically attacks the system, causing physical damage and loss of security protection.

3.6.16 Social engineering to steal password

DA.Hack_SocEng_Password A hacker persuades a user or administrator to reveal his password, giving the hacker access to the person's account privileges.

3.6.17 Hacker uses social engineering to learn system information

DA.Hack_SocEng_SysInfo A hacker persuades a user or administrator to reveal information about system operational procedures, auditing and known flaws.

3.6.18 System hardware fails during system operation

DA.Hardware_Flaw System use uncovers a hardware flaw in a critical system component.

3.6.19 System use uncovers an intrinsic software flaw in a critical system component

DA.Software_Flaw An authorized user performs an operation or set of operations, exercising a software flaw in a security-critical component.

3.6.20 Communications function failure

DA.Failure_DS_Comm Failure of a communications function severs communications between security-critical (TSF) components.

3.6.21 Denial of having received data from another local user

DA.Repudiate_Rcvr_Int A local, authorized user receives a message from another local user via the system, and then denies having received it. This typically affects the sender of the message who is counting on responsibilities associated with receipt of the message.

3.6.22 Denial of having received information from a remote user

DA.Repudiate_Rcvr_Local A local, authorized user receives a message from another user at a remote trusted product, and then denies having received it.

3.6.23 Denial of having received information by a remote user

DA.Repudiate_Rcvr_Rem A local, authorized user sends a message to another user at a remote trusted product who then denies having received it.

3.6.24 Circumvent non-repudiation in a transaction involving a user and a local system

DA.Repudiate_Trans_Loc An authorized user participates in a transaction by responding to system/application prompts and then denies that the dialogue took place. The user and system/application are collocated.

3.6.25 Circumvent non-repudiation in a transaction involving a local user and a remote system

DA.Repudiate_Trans_Uloc An authorized user participates in a transaction by responding to remote system/application prompts and then denies that the dialogue took place.

3.6.26 Circumvent non-repudiation in a transaction involving a remote user and a local system

DA.Repudiate_Trans_Urem An authorized remote user participates in a transaction by responding to local system/application prompts and then denies that the dialogue took place.

3.6.27 User error deletes data

DA.User_Err_Delete An authorized user accidentally deletes user data.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Accidental	Constructive	Zero	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Error

Result

Loss Types	IT Capabilities	Locations
Availability	User Data	TOE

3.6.28 User obstructs legitimate use of resources.

DA.User_Obst_Res_Use An authorized user obstructs the use resources by unauthorized modification of data file, communication channel, or object security attributes.

Agent

Agent Type	Authentication	Attitude	Motive	Sophistication	Localities
Human	Authenticated	Accidental	Negligent	Zero	Any

Method

Lifecycle Phases	Human Role	Action
Operation	No Duties	Modify

Result

Loss Types	Locations
Availability	TOE

3.7 Organizational General Security Policies for the TOE

The following General Policy Statements are addressed directly by the TOE:

3.7.1 Individual accountability

GP.Accountability Individuals shall be held accountable for their actions.

3.7.2 Notification of threats and vulnerabilities

GP.Authorities Appropriate authorities shall be immediately notified of any threats or vulnerabilities impacting systems that process their data.

3.7.3 Authorized use of information

GP.Authorized_Use Information shall be used only for its authorized purpose(s).

3.7.4 Installation and usage guidance

GP.Guidance Guidance shall be provided for the secure installation and use of the system.

3.7.5 System lifecycle phases integrate security

GP.Lifecycle Information systems security shall be an integral part of all system lifecycle phases.

3.7.6 Information marking

GP.Marking Information shall be appropriately marked and labeled.

3.7.7 Semiformally designed, tested and reviewed

GP.Product_Assurance The Developer shall gain maximum assurance from security engineering based upon rigorous commercial development practices supported by moderate application of specialist security engineering techniques. Such a TOE will probably be designed and developed with the intent of achieving this level of assurance. This is applicable in those circumstances where developers or users require a high level of independently assured security in a planned development and require a rigorous development approach without incurring unreasonable costs attributable to specialist security engineering techniques.

3.8 Organizational Detailed Security Policies assigned to the TOE

Each General Policy Statement, as defined in Sec. 3.7 and Sec. 3.9, refines into a number of Detailed Policy Statements, that provide fine-graded policies that can be mapped onto Security Objectives. This section documents the series of Detailed Policy Statements whose compliance is a TOE responsibility.

3.8.1 Changes to security data by authorized personnel

DP.Admin_Security_Data Provide mechanisms to assure that changes to security related data are executed only by authorized personnel.

3.8.2 Configuration Management

DP.Assurance_ACM Configuration management (CM) helps to ensure that the integrity of the TOE is preserved, by requiring discipline and control in the processes of refinement and modification of the TOE and other related information. CM prevents unauthorised modifications, additions, or deletions to the TOE, thus providing assurance that the TOE and documentation used for evaluation are the ones prepared for distribution.

3.8.3 Delivery and Operation

DP.Assurance_ADO This security policy defines requirements for the measures, procedures, and standards concerned with secure delivery, installation, and operational use of the TOE, ensuring that the security protection offered by the TOE is not compromised during transfer, installation, start-up, and operation.

3.8.4 Product Development

DP.Assurance_ADV This security policy defines requirements for the stepwise refinement of the TSF from the TOE summary specification in the ST down to the actual implementation. Each of the resulting TSF representations provide information to help the evaluator determine whether the functional requirements of the TOE have been met.

3.8.5 Guidance documents

DP.Assurance_AGD This security policy defines requirements directed at the understandability, coverage and completeness of the operational documentation provided by the developer. This documentation, which provides two categories of information, for users and for administrators, is an important factor in the secure operation of the TOE.

3.8.6 Lifecycle support

DP.Assurance_ALC This security policy defines requirements for assurance through the adoption of a well defined life-cycle model for all the steps of the TOE development, including flaw remediation procedures and policies, correct use of tools and techniques and the security measures used to protect the development environment.

3.8.7 Test

DP.Assurance_ATE This security policy states testing requirements that demonstrate that the TSF satisfies the TOE security functional requirements.

3.8.8 Vulnerability Assesment

DP.Assurance_AVA This security policy defines requirements directed at the identification of exploitable vulnerabilities. Specifically, it addresses those vulnerabilities introduced in the construction, operation, misuse, or incorrect configuration of the TOE.

3.8.9 Individual accountability

DP.Audit_Gen_User The system shall provide individual accountability for auditable actions.

3.8.10 Audit data generation with identity

DP.Audit_Generation The system shall provide the capability to ensure that all audit records include enough information to determine the date and time of action, the system locale of the action, the system entity that initiated or completed the action, the resources involved, and the action involved.

3.8.11 Protected audit data storage

DP.Audit_Protect The system shall protect the contents of the audit trails against unauthorized access, modification, or deletion.

3.8.12 Notification of threats and vulnerabilities

DP.Authority_Notify Notification of threats and vulnerabilities shall be addressed.

3.8.13 Notification of data content changes

DP.Change_Control_Users Notify user of the time and date of the last modification of data.

3.8.14 Implement operational configuration management

DP.Config_Mgt_Plan A configuration management plan shall be implemented by the system. The system shall implement configuration management to assure storage integrity, identification of system connectivity (software, hardware, and firmware), and identification of system components (software, hardware, and firmware). The system shall implement strong integrity mechanisms (integrity locks, encryption).

3.8.15 Documented recovery

DP.Documented_Recovery The system shall provide procedures and features to assure that system recovery is done in a trusted and secure manner. Any circumstances that could result in an untrusted recovery shall be documented.

3.8.16 Labeling data

DP.External_Labels The system shall provide security parameters associated with information exchanged between systems.

3.8.17 User identification and authentication

DP.I&A_User The system shall provide Identification and authentication procedures which uniquely identify and authenticate users.

3.8.18 Strong integrity mechanisms

DP.Integrity_Data/SW The system shall implement strong integrity mechanisms (integrity locks, encryption).

3.8.19 Operational integrity system function testing

DP.Integrity_Practice Provide system functional tests to periodically test the integrity of the hardware and code running system functions.

3.8.20 Security throughout lifecycle

DP.Lifecycle_Security Security shall be addressed throughout the system's lifecycle.

3.8.21 Privileged user access

DP.Need_To_Know The system shall function so that each user has access to all of the information and functions that the user requires to perform duties, but no more.

3.8.22 Privileged user documentation

DP.Privileged_Doc Documentation shall include guides or manuals for the system's privileged users.

3.8.23 User screen locking

DP.Screen_Lock The system shall provide a screen lock mechanism.

3.8.24 Assurance of effective storage integrity

DP.Storage_Integrity The system shall provide assurance that storage integrity is effective.

3.8.25 System access banners

DP.Sys_Access_Banners The system shall notify users prior to gaining access that the user's actions may be monitored and recorded, that using the system consents to such monitoring, and that unauthorized use may result in criminal or civil penalties.

3.8.26 Validation of security function integrity

DP.Sys_Assur_HW/SW/FW Features and procedures to validate the integrity and the expected operation of the security-relevant software, hardware, and firmware shall be provided by the system.

3.8.27 System backup procedures

DP.Sys_Backup_Procs Provide the capability to restore the system to a secure state after discontinuities of system operations.

3.8.28 Restoration with minimal loss

DP.Sys_Backup_Restore The system shall provide backup procedures to allow restoration of the system with minimal loss of service or data.

3.8.29 Effective backup restoration

DP.Sys_Backup_Storage The system shall provide procedures to ensure both the existence of sufficient backup storage capability and effective restoration (incremental and complete) of the backup data.

3.8.30 Protection from security function modification

DP.System_Protection Provide features or procedures for protection of the system from improper changes.

3.8.31 Trusted system recovery

DP.System_Recovery Provide procedures and features to assure that system recovery is done in a trusted and secure manner.

3.8.32 Encryption of transmitted user data

DP.User_Data_Dial-in The system shall provide data transmission using an encryption mechanism appropriate for the sensitivity of the data.

3.8.33 Protection of stored user data

DP.User_Data_Storage The system shall provide appropriate storage, continuous personnel access control storage, or encrypted storage of data based on the sensitivity of the data.

3.8.34 Protection of transmitted user data

DP.User_Data_Transfer The system shall provide a protected distribution system for data transmitted.

3.8.35 Discretionary access control

DP.User_Defined_AC The system shall provide a Discretionary Access Control (DAC) function (i.e., a user can grant access authorization to other users for data they control).

3.8.36 General user documentation

DP.User_Documentation Documentation shall include a user's guide for the general user.

3.9 Organizational General Security Policies for the TOE and the Environment

The following General Policy Statements are addressed by the TOE and the Environment as defined in the Assumptions:

3.9.1 Information availability

GP.Availability Information shall be available to satisfy mission requirements.

3.9.2 Information access control

GP.Information_AC Information shall be accessed only by authorized individuals and processes.

3.9.3 Information content integrity

GP.Integrity Information shall retain its content integrity.

3.9.4 Physical protection

GP.Physical_Control Information shall be physically protected to prevent unauthorized disclosure, destruction, or modification.

3.10 Organizational Detailed Security Policies assigned to the Environment

Each General Policy Statement, as defined in Sec. 3.9, refines into a number of Detailed Policy Statements, that provide fine-graded policies that can be mapped onto Security Objectives. This section documents the series of Detailed Policy Statements whose compliance is a responsibility assumed for the environment.

3.10.1 Malicious code prevention

DP.Malicious_Code Procedures and mechanisms to prevent the introduction of malicious code into the system shall be provided.

3.10.2 Backup protection and restoration

DP.Sys_Backup_Verify The system shall provide appropriate physical and technical protection of the backup and restoration hardware, firmware, and software.

3.10.3 Enhanced user identification and authentication

DP.User_Auth_Enhanced The system shall require the use of enhanced authentication for privileged users who either reside outside of the system's perimeter or whose communications traverse data lines outside of the system's perimeter.

3.10.4 Non-repudiation capabilities

DP.Non-Repudiation The system shall provide non-repudiation capabilities.

3.10.5 Physical tampering detection and notification

DP.Tamper_ID The system shall detect physical tampering and notify the appropriate authority.

Chapter 4

SECURITY OBJECTIVES

Security objectives cover two complementary security needs, those oriented to counter the identified threats and attacks and those aimed to fulfill the organizational policies.

4.1 Security Objectives for the TOE

These are established to counter the identified threats and applicable policies of the defined TOE:

4.1.1 Limitation of administrative access control

O.AC_Admin_Limit Design administrative functions in such a way that administrators do not automatically have access to user objects, except for necessary exceptions. For an accounts administrator, the necessary exceptions include allocation and de-allocation of storage. For an audit administrator, the necessary exceptions include observation of audited actions. In general, the exceptions tend to be role specific.

4.1.2 Object security attributes and exportation

O.AC_Label_Export Provide object security attributes in exported data with moderate to high effectiveness. The attributes are those associated with specific security function policies.

4.1.3 Access history for user session

O.Access_History Display information related to the most recent successful and unsuccessful attempts to establish a user session, once a user successfully establishes a user session.

4.1.4 Limit an administrator's ability to modify user-subject bindings

O.Adm_Limits_Bindings Limit the administrator from modification of user-subject bindings in an effort to deter users acting without accountability.

4.1.5 Limit administrator's modification of user attributes

O.Adm_User_Att_Mod Deter the administrator from maliciously modifying users' attributes. Such modifications could allow unauthorized user actions or denial of service to a legitimate user.

4.1.6 Administrative validation of executables

O.Admin_Code_Val Validate executable objects prior to allowing execution. Validation needs to be done by someone with an expertise to recognize malicious code and the authority and means to prevent its execution.

4.1.7 Software validation for absence of steganography

O.Admin_Code_Val_Sten Validate exported objects for absence of steganographic content prior to allowing exportation. Validation needs to be done by someone with an expertise to recognize hidden content and the authority and means to prevent its export.

4.1.8 Administrator guidance documentation

O.Admin_Guidance Deter administrator errors by providing adequate administrator guidance.

4.1.9 Apply patches to fix the code

O.Apply_Code_Fixes Apply patches to fix the code when vulnerabilities in code allow unauthorized and undiscovered access.

4.1.10 CM automation

O.Assurance_ACM_AUT The objective of introducing automated CM tools is to increase the effectiveness of the CM system. While both automated and manual CM systems can be bypassed, ignored, or prove insufficient to prevent unauthorised modification, automated systems are less susceptible to human error or negligence.

4.1.11 CM capabilities

O.Assurance_ACM_CAP The capabilities of the CM system address the likelihood that accidental or unauthorised modifications of the configuration items will occur. The CM system should ensure the integrity of the TOE from the early design stages through all subsequent maintenance efforts.

4.1.12 CM scope

O.Assurance_ACM_SCP The objective of this family is to ensure that all necessary TOE configuration items are tracked by the CM system. This helps to ensure that the integrity of these configuration items is protected through the capabilities of the CM system.

4.1.13 Delivery

O.Assurance_ADO_DEL The requirements for delivery call for system control and distribution facilities and procedures that provide assurance that the recipient receives the TOE that the sender intended to send, without any modifications. For a valid delivery, what is received must correspond precisely to the TOE master copy, thus avoiding any tampering with the actual version, or substitution of a false version.

4.1.14 Installation, generation and start-up

O.Assurance_ADO_IGS Installation, generation, and start-up procedures are useful for ensuring that the TOE has been installed, generated, and started up in a secure manner as intended by the developer. The requirements for installation, generation and start-up call for a secure transition from the TOE's implementation representation being under configuration control to its initial operation in the user environment.

4.1.15 Functional specification

O.Assurance_ADV_FSP The functional specification is a high-level description of the user-visible interface and behaviour of the TSF. It is an instantiation of the TOE security functional requirements. The functional specification has to show that all the TOE security functional requirements are addressed.

4.1.16 High-level design

O.Assurance_ADV_HLD The high-level design of a TOE provides a description of the TSF in terms of major structural units (i.e. subsystems) and 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 TOE security functional requirements.
The high-level design refines the functional specification into subsystems. For each subsystem of the TSF, the high-level design describes its purpose and function, and identifies the security functions contained in the subsystem. The interrelationships of all subsystems are also defined in the high-level design. These interrelationships will be represented as external interfaces for data flow, control flow, etc., as appropriate.

4.1.17 Implementation representation

O.Assurance_ADV_IMP The description of the implementation representation in the form of source code, firmware, hardware drawings, etc. captures the detailed internal workings of the TSF in support of analysis.

4.1.18 TSF internals

O.Assurance_ADV_INT This family addresses the internal structure of the TSF. Requirements are presented for modularity, layering (to separate levels of abstraction and minimise circular dependencies), minimisation of the complexity of policy enforcement mechanisms, and the minimisation of the amount of non-TSP-enforcing functionality within the TSF - thus resulting in a TSF that is simple enough to be analysed.
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. Thus, a modular design provides the basis for determining the scope of interaction with other elements of the TSF, provides for increased assurance that unexpected effects do not occur, and also provides the basis for designing and evaluating test suites.

4.1.19 Low-level design

O.Assurance_ADV_LLD The low-level design of a TOE provides a description of the internal workings of the TSF in terms of modules and their interrelationships and dependencies. The low-level design provides assurance that the TSF subsystems have been correctly and effectively refined.
For each module of the TSF, the low-level design describes its purpose, function, interfaces, dependencies, and the implementation of any TSP-enforcing functions.

4.1.20 Representation correspondence

O.Assurance_ADV_RCR The correspondence between the various TSF representations (i.e. TOE summary specification, functional specification, high-level design, low-level design, implementation representation) addresses the correct and complete instantiation of the requirements to the least abstract TSF representation provided. This conclusion is achieved by step-wise refinement and the cumulative results of correspondence determinations between all adjacent abstractions of representation.

4.1.21 Security policy modeling

O.Assurance_ADV_SPM It is the objective of this family to provide additional assurance that the security functions in the functional specification enforce the policies in the TSP. This is accomplished via the development of a security policy model that is based on a subset of the policies of the TSP, and establishing a correspondence between the functional specification, the security policy model, and these policies of the TSP.

4.1.22 Administrator guidance

O.Assurance_AGD_ADM Administrator guidance refers to written material that is intended to be used by those persons responsible for configuring, maintaining, and administering the TOE in a correct manner for maximum security. Because the secure operation of the TOE is dependent upon the correct performance of the TSF, persons responsible for performing these functions are trusted by the TSF. Administrator guidance is intended to help administrators understand the security functions provided by the TOE, including both those functions that require the administrator to perform security-critical actions and those functions that provide security-critical information.

4.1.23 User guidance

O.Assurance_AGD_USR User guidance refers to material that is intended to be used by non-administrative human users of the TOE, and by others (e.g. programmers) using the TOE's external interfaces. User guidance describes the security functions provided by the TSF and provides instructions and guidelines, including warnings, for its secure use.

4.1.24 Development security

O.Assurance_ALC_DVS Development security is concerned with physical, procedural, personnel, and other security measures that may be used in the development environment to protect the TOE. It includes the physical security of the development location and any procedures used to select development staff.

4.1.25 Flaw remediation

O.Assurance_ALC_FLR Flaw remediation requires that discovered security flaws be tracked and corrected by the developer. Although future compliance with flaw remediation procedures cannot be determined at the time of the TOE evaluation, it is possible to evaluate the policies and procedures that a developer has in place to track and correct flaws, and to distribute the flaw information and corrections.

4.1.26 Life cycle definition

O.Assurance_ALC_LCD Poorly controlled development and maintenance of the TOE can result in a flawed implementation of a TOE (or a TOE that does not meet all of its security

requirements). This, in turn, results in security violations. Therefore, it is important that a model for the development and maintenance of a TOE be established as early as possible in the TOE's life-cycle.
Using a model for the development and maintenance of a TOE does not guarantee that the TOE will be free of flaws, nor does it guarantee that the TOE will meet all of its security functional requirements. It is possible that the model chosen will be insufficient or inadequate and therefore no benefits in the quality of the TOE can be observed. Using a life-cycle model that has been approved by some group of experts (e.g. academic experts, standards bodies) improves the chances that the development and maintenance models will contribute to the overall quality of the TOE.

4.1.27 Tools and techniques

O.Assurance_ALC_TAT Tools and techniques is an aspect of selecting tools that are used to develop, analyse and implement the TOE. It includes requirements to prevent ill-defined, inconsistent or incorrect development tools from being used to develop the TOE. This includes, but is not limited to, programming languages, documentation, implementation standards, and other parts of the TOE such as supporting runtime libraries.

4.1.28 Coverage

O.Assurance_ATE_COV This family addresses those aspects of testing that deal with completeness of test coverage. That is, it addresses the extent to which the TSF is tested, and whether or not the testing is sufficiently extensive to demonstrate that the TSF operates as specified.

4.1.29 Depth

O.Assurance_ATE_DPT The components in this family deal with the level of detail to which the TSF is tested. Testing of security functions is based upon increasing depth of information derived from analysis of the representations.
The objective is to counter the risk of missing an error in the development of the TOE. Additionally, the components of this family, especially as testing is more concerned with the internal structure of the TSF, are more likely to discover any malicious code that has been inserted.
Testing that exercises specific internal interfaces can provide assurance not only that the TSF exhibits the desired external security behaviour, but also that this behaviour stems from correctly operating internal mechanisms.

4.1.30 Functional tests

O.Assurance_ATE_FUN Functional testing performed by the developer establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of its PP/ST. Such functional testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. The family "Functional tests" is focused on the type and amount of documentation or support tools required, and what is to be demonstrated through developer testing. Functional testing is not limited to positive confirmation that the required security functions are provided, but may also include negative testing to check for the absence of particular undesired behaviour (often based on the inversion of functional requirements).

4.1.31 Independent testing

O.Assurance_ATE_IND One objective is to demonstrate that the security functions perform as specified.

An additional objective is to counter the risk of an incorrect assessment of the test outcomes on the part of the developer that results in the incorrect implementation of the specifications, or overlooks code that is non-compliant with the specifications.

4.1.32 Covert channel analysis

O.Assurance_AVA_CCA Covert channel analysis is carried out to determine the existence and potential capacity of unintended signalling channels (i.e. illicit information flows) that may be exploited.

The assurance requirements address the threat that unintended and exploitable signalling paths exist that may be exercised to violate the SFP.

4.1.33 Misuse

O.Assurance_AVA_MSU Misuse investigates whether the TOE can be configured or used in a manner that is insecure but that an administrator or user of the TOE would reasonably believe to be secure.

4.1.34 Strength of TOE security functions

O.Assurance_AVA_SOF Even if a TOE security function cannot be bypassed, deactivated, or corrupted, it may still be possible to defeat it because there is a vulnerability in the concept of its underlying security mechanisms. For those functions a qualification of their security behaviour can be made using the results of a quantitative or statistical analysis of the security behaviour of these mechanisms and the effort required to overcome them. The qualification is made in the form of a strength of TOE security function claim.

4.1.35 Vulnerability analysis

O.Assurance_AVA_VLA Vulnerability analysis is an assessment to determine whether vulnerabilities identified, during the evaluation of the construction and anticipated operation of the TOE or by other methods (e.g. by flaw hypotheses), could allow users to violate the TSP.

4.1.36 Complete security functions or recover to previous state

O.Atomic_Functions Recover automatically to a consistent, secure state if a security function does not complete successfully in the presence of certain types of failures.

4.1.37 Audit changes of system entry parameters

O.Aud_Sys_Entry_Parms Deter an administrator from changing system entry parameters to allow an unauthorized user access to organizational assets to which they are forbidden.

4.1.38 Auditing for user accountability

O.Audit_Account Provide information about past user behavior to an authorized user through system mechanisms. Specifically, during any specified time interval, the system is able to report to a user acting in an identified audit role selected auditable actions that a user has performed, and as a result, what auditable objects were affected and what auditable information was received by that user.

4.1.39 Audit-administration role duties

O.Audit_Admin_Role Deter modification or destruction of audit data through the creation of an audit-administration role.

4.1.40 Audit system access to deter misuse

O.Audit_Deter_Misuse Audit system access to discover system misuse and provide a potential deterrent by warning the user.

4.1.41 Individual accountability

O.Audit_Gen_User Provide individual accountability for audited events. Uniquely identify each user so that auditable actions can be traced to a user.

4.1.42 Audit records with identity

O.Audit_Generation Record in audit records: date and time of action, location of the action, and the entity responsible for the action.

4.1.43 Respond to possible loss of stored audit records

O.Audit_Loss_Respond Respond to possible loss of audit records when audit trail storage is full or nearly full.

4.1.44 Protect stored audit records

O.Audit_Protect Protect audit records against unauthorized access, modification, or deletion to ensure accountability of user actions.

4.1.45 User notification of data content changes

O.Change_Control_Users Notify users of changes to data content in order to make any adjustments to their own data.

4.1.46 Code signing and verification

O.Code_Signing Check verification of signed downloaded code prior to execution. A well-known example is checking digital signatures on signed Java applets.

4.1.47 Trusted channel to remote trusted system

O.Comm_Trusted_Channel Provide a communications channel between the system and a remote trusted system for the performance of security-critical operations.

4.1.48 Implement operational configuration management

O.Config_Management Implement a configuration management plan. Implement configuration management to assure storage integrity, identification of system connectivity (software, hardware, and firmware), and identification of system components (software, hardware, and firmware).

4.1.49 Verify correct operation as designed

O.Correct_Operation Provide the ability for the authorized user to verify that the system operates as designed.

4.1.50 Cryptographic access control policy

O.Crypto_AC Restrict user access to cryptographic IT assets in accordance with a specified user access control policy.

4.1.51 Encrypted communications channel

O.Crypto_Comm_Channel Provide secure session establishment between the system and remote systems using encryption functions.

4.1.52 Separation of cryptographic data

O.Crypto_Data_Sep Provide complete separation between plaintext and encrypted data and between data and keys. This requires separate channels and separate storage areas. The only place any data can pass between the plaintext and encrypted data modules is in the cryptographic engine. There should be no way for plaintext keys to reach either data module and no way for data to enter the key handling module. Encrypted keys can be handled as encrypted data, but with limited user access.

4.1.53 Cryptographic Design and Implementation

O.Crypto_Dsgn_Impl Minimize or even eliminate design and implementation errors in the cryptographic modules and functions.

4.1.54 Cryptographic import, export, and inter-TSF transfer

O.Crypto_Import_Export Protect cryptographic data assets when they are being transmitted to and from the TOE, either through intervening untrusted components or directly to/from human users.

4.1.55 Cryptographic Key Management

O.Crypto_Key_Man Fully define cryptographic components, functions, and interfaces. Ensure appropriate protection for cryptographic keys throughout their lifecycle, covering generation, distribution, storage, use, and destruction.

4.1.56 Management of cryptographic roles

O.Crypto_Manage_Roles Provide one or more roles to manage cryptographic assets and attributes.

4.1.57 Cryptographic Modular Design

O.Crypto_Modular_Dsgn Prevent errors in one part of the TOE from influencing other parts, especially cryptographic parts. To this end, noncryptographic I/O paths must be well defined and logically independent of circuitry and processes performing key generation, manual key entry, key zeroising, and similar key-related operations.

4.1.58 Cryptographic function definition

O.Crypto_Operation Cryptographic components, functions, and interfaces shall be fully defined.

4.1.59 Cryptographic self test

O.Crypto_Self_Test Provide the ability to verify that the cryptographic functions operate as designed.

4.1.60 Test cryptographic functionality

O.Crypto_Test_Reqs Test cryptographic operation and key management.

4.1.61 Enforce data exchange confidentiality

O.Data_Exchange_Conf Protect user data confidentiality when exchanging data with a remote system.

4.1.62 Control user data exportation

O.Data_Export_Control Impose information control policies that do not allow export of specified data and/or export to specified locations.

4.1.63 Data import/export to/from system control

O.Data_Imp_Exp_Control Protect data from being sent to erroneous places and more places external to the system than allowed by the organization's security policy. Conversely the import of data into the system should be protected from illicit information or information not allowed by the organization's security policy.

4.1.64 Sanitize data objects containing hidden or unused data

O.Export_Control Sanitize data objects that may contain hidden data when they are exported from the TOE in order to inhibit steganographic smuggling.

4.1.65 Label or mark information for external systems

O.External_Labels Label or mark information for external systems to prevent the exchange of inappropriate data between systems.

4.1.66 Preservation of secure state for failures in critical components

O.Fail_Secure Preserve the secure state of the system in the event of a secure component failure.

4.1.67 Periodically check integrity

O.General_Integ_Checks Provide periodic integrity checks on both system and user data.

4.1.68 Guarantee the availability of audit storage space

O.Guarantee_Audit_Stg Maintain audit data and guarantee space for that data.

4.1.69 Limit sessions to outside users

O.Hack_Limit_Sessions Limit the number of sessions available to outside users. A hacker can initiate multiple communication sessions that could cause an overload on resources, for example, half open session starts as is seen in SYN flood attacks.

4.1.70 Control hacker communication traffic

O.Hack_Traffic_Control Control (e.g. reroute or discard) hacker communication traffic to prevent potential damage.

4.1.71 Identify and authenticate a user to support accountability

O.I&A_Domain Provide the basic I&A functions that will support user accountability.

4.1.72 Transaction identification and authentication

O.I&A_Transaction Associate each transaction between a user and a system/application with a unique transaction ID, allowing events associated with a given transaction to be distinguished from other events involving the user and/or system/application.

4.1.73 Identify and authenticate each user

O.I&A_User Uniquely identify and authenticate each user of the system.

4.1.74 User-action identification and authentication

O.I&A_User_Action Associate each user-requested action with the user who requested the action.

4.1.75 Identify unusual user activity

O.Identify_Unusual_Act Identify unusual user activity on the system.

4.1.76 System enforced information flow

O.Info_Flow_Control Enforce an information flow policy whereby users are constrained from allowing access to information they control, regardless of their intent (e.g., mandatory access control). This lattice property of security attributes is commonly associated with the U.S. DoD implementations of Mandatory Access Control (MAC).

4.1.77 Provide information flow control administration

O.Info_Flow_Ctrl_Admin Manage information flow control policy and functions to allow only specified administrators to have the ability to manipulate the information flow control.

4.1.78 Require inspection for absence of malicious code.

O.Input_Inspection Require inspection of downloads/transfers.

4.1.79 Data marking integrity export

O.Integ_Data_Mark_Exp Ensure that data markings are included with data that is exported to another trusted product.

4.1.80 Integrity of system data transferred externally

O.Integ_Sys_Data_Ext Ensure the integrity of system data exchanged externally with another trusted product by using a protocol for data transfer that will permit error detection and correction. This includes detecting and possibly correcting errors in data received and encoding outgoing data to make it possible for the receiver to detect and possibly correct errors. The method for detecting and correcting errors is based on some method (protocol) that is agreed upon by participating parties.

4.1.81 Integrity of system data transferred internally

O.Integ_Sys_Data_Int Ensure the integrity of system data transferred internally.

4.1.82 Protect user data during internal transfer

O.Integ_User_Data_Int Ensure the integrity of user data transferred internally within the system.

4.1.83 Correct attribute exchange with another trusted product

O.Integrity_Attr_Exch Ensure that the system correctly exchanges security-attribute information with another trusted IT product.

4.1.84 Integrity protection for user data and software

O.Integrity_Data/SW Provide integrity protection for user data and software.

4.1.85 Integrity of system data replication

O.Integrity_Data_Rep Ensure that when system data replication occurs across the system the data is consistent for each replication.

4.1.86 Operational integrity system function testing

O.Integrity_Practice Provide system functional tests to periodically test the integrity of the hardware and code running system functions.

4.1.87 Isolate untrusted executables

O.Isolate_Executables Run executable code in a protected domain where the code's potential errors or malicious code will not significantly impact other system functions of other valid users of the system.

4.1.88 Lifecycle security

O.Lifecycle_Security Provide tools, techniques, and security employed during the development phase. Detect and resolve flaws during the operational phase. Provide safe destruction techniques.

4.1.89 Restrict actions before authentication

O.Limit_Actions_Auth Restrict the actions a user may perform before the TOE verifies the identity of the user.

4.1.90 Limit the number of user initiated communication sessions

O.Limit_Comm_Sessions Provide mechanisms to limit the number of sessions that the user can initiate, if the user initiates multiple sessions that exceed the processors ability to perform in a reliable and efficient manner. These sessions could ether be communication (TCP/IP) sessions or user login sessions.

4.1.91 Limit multiple sessions

O.Limit_Mult_Sessions Provide the capability to limit the number of sessions that a user may have open at one time.

4.1.92 Maintain security domain

O.Maintain_Sec_Domain Maintain at least one security domain for system (TOE) execution to protect the TOE from interference and tampering.

4.1.93 Controlled access by maintenance personnel

O.Maintenance_Access Control access to the system by maintenance personnel who troubleshoot the system and perform system updates.

4.1.94 Expiration of maintenance privileges

O.Maintenance_Recover Terminate maintenance user system access privilege automatically after expiration of assigned timed interval.

4.1.95 Manage resource security attributes

O.Manage_Res_Sec_Attr Provide management on resource security attributes.

4.1.96 Manage security-critical data to avoid storage space being exceeded

O.Manage_TSF_Data Manage security-critical (TSF) data to ensure that the size of the data does not exceed the space allocated for storage of the data.

4.1.97 Eliminate residual information

O.No_Residual_Info Ensure there is no object reuse; i.e., ensure that there is no residual information in some information containers or system resources upon their reallocation to different users.

4.1.98 Non-repudiation support for sent information by the nonlocal receiving TSF.

O.NonRepud_Assess_Sent Support nonrepudiation for sent information by supporting remote handling of nonrepudiation evidence if needed.

4.1.99 Non-repudiation support for sent information by the sender's TSF.

O.NonRepud_Gen_Sent Prevent a user from avoiding accountability for sending a message to a recipient at a different site by providing evidence that the user sent the message.

4.1.100 Non-repudiation for sent information, local users

O.NonRepud_Locals_Sent Prevent user from avoiding accountability for sending a message to another user on the same system by providing evidence that the user sent the message.

4.1.101 Non-repudiation for sent information

O.NonRepudiate_Sent Provide evidence that a user sent information.

4.1.102 Basic object attribute integrity

O.Obj_Attr_Integrity Maintain object security attributes with moderate to high accuracy (under the guidance of qualified users).

4.1.103 Object domain protection

O.Obj_Protection Require domain protection for objects. Specify object classes (domains), user groups, and operation classes. Use these to specify which operations may be performed on which objects by which users. Basically this controls what users can do in a given group.

4.1.104 Provide priority of service

O.Priority_Of_Service Control access to resources so that lower-priority activities do not unduly interfere with or delay higher-priority activities.

4.1.105 Privileged-interface status

O.Prvlg_IF_Status Provide capability for an administrator to determine the use status of all privileged interfaces. This would include interfaces used by maintenance personnel.

4.1.106 Identify message modification in messages received

O.Rcv_MsgMod_ID The TSF recognizes changes to messages that occurred in transit, including insertion of spurious messages and deletion or replay of legitimate messages.

4.1.107 Recovery from modification of received messages

O.Rcv_MsgMod_Rcvr The TSF detects and corrects changes in messages received from a remote trusted site.

4.1.108 Provide reference monitor

O.Reference_Monitor Always invoke mechanisms that enforce security policies (i.e., as for a traditional reference monitor).

4.1.109 Disable remote execution

O.Remote_Execution Disable a remote entity's ability to execute local code.

4.1.110 Resource quotas for users and services

O.Resource_Quotas Use resource quotas to limit user and service use of system resources to a level that will prevent degradation or denial of service to other critical users and services.

4.1.111 User screen locking

O.Screen_Lock Provide a screen lock function to prevent an unauthorized user from using an unattended computer where a valid user has an active session.

4.1.112 Security-relevant configuration management

O.Secure_Configuration Manage and update system security policy data and enforcement functions, and other security-relevant configuration data, in accordance with organizational security policies.

4.1.113 Protect and maintain secure system state

O.Secure_State Maintain and recover to a secure state without security compromise after system error or other interruption of system operation.

4.1.114 Manage security attributes

O.Security_Attr_Mgt Manage the initialization of, values for, and allowable operations on security attributes.

4.1.115 Manage security-critical data

O.Security_Data_Mgt Manage the initialization of, limits on, and allowable operations on security-critical data.

4.1.116 Manage behavior of security functions

O.Security_Func_Mgt Provide management mechanisms for security mechanisms.

4.1.117 Security roles

O.Security_Roles Maintain security-relevant roles and the association of users with those roles.

4.1.118 System terminates session for inactivity

O.Session_Termination System terminates a session after a given interval of inactivity.

4.1.119 Identify message modification in messages sent

O.Snt_MsgMod_ID The TSF supports recognition of changes to transmitted messages that occurred in transit, including insertion of spurious messages and deletion or replay of legitimate messages.

4.1.120 Support recovery from modification of sent messages

O.Snt_MsgMod_Rcvr The TSF supports detection of changes in messages sent to a remote trusted site.

4.1.121 Examine the source code for developer flaws

O.Source_Code_Exam Examine for accidental or deliberate flaws in code made by the developer. The accidental flaws could be lack of engineering detail or bad design. Where the deliberate flaws would include building trapdoors for later entry as an example.

4.1.122 Storage integrity

O.Storage_Integrity Provide integrity for data.

4.1.123 System access banners

O.Sys_Access_Banners Inform the user of the possibility of the system monitoring his actions, and that misuse of the system may result in criminal or civil penalties.

4.1.124 Validation of security function

O.Sys_Assur_HW/SW/FW Ensure that security-relevant software, hardware, and firmware are correctly functioning through features and procedures.

4.1.125 System backup procedures

O.Sys_Backup_Procs Provide backup procedures to ensure that the system can be reconstructed.

4.1.126 Frequent backups to prevent minimal loss

O.Sys_Backup_Restore Provide through frequent backups, restoration of security-relevant changes to the system between backup and restore, and restoration of the security-relevant system state (e.g. access control list) without destruction of other system data.

4.1.127 Sufficient backup storage and effective restoration

O.Sys_Backup_Storage Provide sufficient backup storage and effective restoration to ensure that the system can be recreated.

4.1.128 Protection of system security function

O.Sys_Self_Protection Protect the system security functions through technical features.

4.1.129 Limit administrator's modification of security-critical code or data

O.TSF_Mod_Limit Limit the malicious modification of security-critical (TSF) code and data to include specific system code to prevent the system security protection capabilities from being diminished or weakened.

4.1.130 Local detection of received security-critical data modified in transit

O.TSF_Rcv_Err_ID_Loc Identification by the system (TOE) of modification of security-critical (TSF) data occurring in transit from a remote trusted site must occur.

4.1.131 Remote detection of received security-critical data modified in transit

O.TSF_Rcv_Err_ID_Rem Identification by the remote site of the modification of security-critical (TSF) data occurring in transit from the remote site must occur.

4.1.132 Local detection of sent security-critical data modified in transit

O.TSF_Snd_Err_ID_Loc Identification of modification of security-critical (TSF) data occurring in transit to a remote site by the TSF must occur.

4.1.133 Remote detection of sent security-critical data modified in transit.

O.TSF_Snd_Err_ID_Rem Identification of modification of security-critical (TSF) data occurring in transit to a remote site by the remote site must occur.

4.1.134 Trusted distributed system recovery

O.Trusted_DS_Recov Ensure that a replaced failed component when re-integrated into the system will recover such that it will not cause errors or security breaches in other parts of the system.

4.1.135 Provide a trusted path

O.Trusted_Path Provide a trusted path between the user and the system. Execution of a user-requested action must be made via a trusted path with the following properties:

- The path is logically distinct from, and cannot be confused with other communication paths (by either the user or the system).
- The path provides assured identification of its end points.

4.1.136 Trusted path and channel

O.Trusted_Path&Channel Provide a trusted path to security-critical (TSF) data in which both end points have assured identities. For the remote user, there needs to be a trusted channel as well.

4.1.137 Trusted recovery of security functionality

O.Trusted_Recovery Recovery to a secure state, without security compromise, after a discontinuity of operations.

4.1.138 Documentation of untrusted data recovery

O.Trusted_Recovery_Doc Provide trusted recovery to ensure that data cannot be lost or misplaced. Any circumstances which can cause untrusted recovery to be documented with mitigating procedures established.

4.1.139 Maintain user attributes

O.User_Attributes Maintain a set of security attributes (which may include group membership, clearance, access rights, etc.) associated with individual users in addition to user identity.

4.1.140 User authorization management

O.User_Auth_Management Manage and update user authorization and privilege data in accordance with organizational security and personnel policies.

4.1.141 Basic confidentiality-breach prevention

O.User_Conf_Prevention Prevent unauthorized export of confidential information from the TOE with moderate effectiveness.

4.1.142 Protection of user-session data

O.User_Data_Dial-in Allow dial-in access through secure mechanisms only.

4.1.143 Integrity protection of stored user data

O.User_Data_Integrity Provide appropriate integrity protection for stored user data.

4.1.144 Protection of transmitted user data

O.User_Data_Transfer Provide the ability to have physically protected communications lines, intrusion detection for communications lines, and/or need-to-know isolation for communications lines.

4.1.145 User-defined access control

O.User_Defined_AC Enforce an access control policy whereby users may determine who may access information they control.

4.1.146 User guidance documentation

O.User_Guidance Provide documentation for the general user.

4.2 Security Objectives for the Environment

These are established to counter the identified threats and applicable policies of the assumed TOE environment:

4.2.1 Audit unusual user activity

O.Audit_Unusual_User Audit unusual user activity.

4.2.2 Object and data recovery free from malicious code

O.Clean_Obj_Recovery Recover to a viable state after malicious code is introduced and damage occurs, removing the malicious code as part of the process.

4.2.3 Physical protection of the communications line

O.Comm_Line_Protection Protect communications lines from physical tampering.

4.2.4 Provide fault tolerant operations for critical components

O.Fault_Tolerance Provide fault tolerant operations for critical components and continue to operate in the presence of specific failures in one or more system components.

4.2.5 Limit observation of service usage to authorized users

O.Limit_ObserveRoles Provide authorized users with the capability to observe the usage of specified services or resources as necessary to perform their duties.

4.2.6 Procedures for preventing malicious code

O.Malicious_Code Incorporate malicious code prevention procedures and mechanisms.

4.2.7 Non-repudiation support for received information by a non-local sender's TSF

O.NonRepud_Assess_Recd Support nonrepudiation for received information by supporting remote handling of nonrepudiation evidence if needed.

4.2.8 Non-repudiation support for received information by the recipient's TSF

O.NonRepud_Gen_Recd Prevent a receiving user from avoiding accountability for receiving a message by providing evidence that the user received the message.

4.2.9 Non-repudiation for received information, local users

O.NonRepud_Locals_Rcvd Prevent user from avoiding accountability for receiving a message from another user on the same system by providing evidence that the user received the message.

4.2.10 Non-repudiation for received information

O.NonRepudiate_Recd Provide evidence that a user received information.

4.2.11 Permit users to use services under aliases

O.Permit_Aliases Permit some users to maintain partial anonymity when using specified services or resources by means of aliases.

4.2.12 Permit users to use services anonymously

O.Permit_Anonymity Permit some users to maintain anonymity when using specified services or resources.

4.2.13 Prevent system from collecting user privacy information

O.Prevent_AskPrivInfo Provide some services or resources to specified users without soliciting from the user information that is relevant to the user's privacy.

4.2.14 Prevent linking of multiple service use

O.Prevent_Link Ensure that a user may make multiple uses of a service or resource without other specified users being able to link these uses together.

4.2.15 Prevent observation of service use

O.Prevent_Observe Ensure that a user may use a service or resource without other specified users being able to observe that the service or resource is being used.

4.2.16 React to discovered attacks

O.React_Discovered_Atk Implement automated notification or other reactions to the TSF-discovered attacks in an effort to identify attacks and to create an attack deterrent.

4.2.17 Rollback

O.Rollback Recover from user operations by undoing some user operations (i.e., rolling back) to restore a previous known state.

4.2.18 Detect modifications of backup hardware, firmware, software

O.Sys_Backup_Verify Detect modifications to backup hardware, firmware, and software.

4.2.19 Tamper detection

O.Tamper_ID Provide system features that detect physical tampering of a system component, and use those features to limit security breaches.

4.2.20 Tamper resistance

O.Tamper_Resistance Prevent or resist physical tampering with specified system devices and components.

4.2.21 Enhanced user authentication

O.User_Auth_Enhanced Execute enhanced measures to ensure that either user authentication data cannot be stolen or when it is stolen, it cannot be used to gain access to the system.

4.2.22 Require multiple authentication mechanisms

O.User_Auth_Multiple Invoke multiple authentication mechanisms, which will provide confidence that the user is who they say they are.

Chapter 5

IT SECURITY REQUIREMENTS

5.1 TOE Security Functional Requirements

The required minimum strength of function level is mandated as "**SOF-basic**", for the functional requirements indicated in this section. With this SOF, the TOE shall be resistant to attackers with low attack potential, and remaining vulnerabilities shall only be exploitable by attacker with moderate or high attack potential.

Note that in the threat and attack analysis, there are attacks countered by the TOE whose attack potential is moderate or high, yet the product is claimed to counter them.

This may seem to contradict the previous SOF statement, whereas both claims hold in that the functionality to counter these attacks is included within the TOE but their assessment shall be performed with an assurance level of EAL4.

5.1.1 FAU - Security audit

Security auditing involves recognising, recording, storing, and analysing information related to security relevant activities (i.e. activities controlled by the TSP). The resulting audit records can be examined to determine which security relevant activities took place and whom (which user) is responsible for them.

FAU_GEN - Security audit data generation

This family defines requirements for recording the occurrence of security relevant events that take place under TSF control. This family identifies the level of auditing, enumerates the types of events that shall be auditable by the TSF, and identifies the minimum set of audit-related information that should be provided within various audit record types.

FAU_GEN.1 Audit data generation

Audit data generation defines the level of auditable events, and specifies the list of data that shall be recorded in each record.

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

- Start-up and shutdown of the audit functions;
- All auditable events for the detailed level of audit, and in particular
  1. Audit events for FAU_SAR.1 (See 5.1.1):

- Reading of information from the audit records.

2. Audit events for FAU_SAR.3 (See 5.1.1):
   - The parameters used for the viewing.
3. Audit events for FAU_SEL.1 (See 5.1.1):
   - All modifications to the audit configuration that occur while the audit collection functions are operating.
4. Audit events for FAU_STG.4 (See 5.1.1):
   - Actions taken due to the audit storage failure.
5. Audit events for FCO_NRO.2 (See 5.1.2):
   - The invocation of the non-repudiation service.
   - Identification of the information, the destination, and a copy of the evidence provided.
   - The identity of the user who requested a verification of the evidence.
6. Audit events for FCS_CKM.1 (See 5.1.3):
   - Success and failure of the activity.
   - The object attribute(s), and object value(s) excluding any sensitive information (e.g. secret or private keys).
7. Audit events for FCS_CKM.2 (See 5.1.3):
   - Success and failure of the activity.
   - The object attribute(s), and object value(s) excluding any sensitive information (e.g. secret or private keys).
8. Audit events for FCS_CKM.3 (See 5.1.3):
   - Success and failure of the activity.
   - The object attribute(s), and object value(s) excluding any sensitive information (e.g. secret or private keys).
9. Audit events for FCS_CKM.4 (See 5.1.3):
   - Success and failure of the activity.
   - The object attribute(s), and object value(s) excluding any sensitive information (e.g. secret or private keys).
10. Audit events for FCS_COP.1 (See 5.1.3):
    - Success and failure, and the type of cryptographic operation.
    - Any applicable cryptographic mode(s) of operation, subject attributes and object attributes.
11. Audit events for FDP_ACF.1 (See 5.1.4):
    - Successful requests to perform an operation on an object covered by the SFP.
    - All requests to perform an operation on an object covered by the SFP.
    - The specific security attributes used in making an access check.
12. Audit events for FDP_DAU.2 (See 5.1.4):
    - Successful generation of validity evidence.
    - Unsuccessful generation of validity evidence.
    - The identity of the subject that requested the evidence.
    - The identity of the subject that generated the evidence.
13. Audit events for FDP_ETC.1 (See 5.1.4):
    - Successful export of information.
    - All attempts to export information.
14. Audit events for FDP_ETC.2 (See 5.1.4):

  - Successful export of information.
  - All attempts to export information.

15. Audit events for FDP_IFF.1 (See 5.1.4):
    - Decisions to permit requested information flows.
    - All decisions on requests for information flow.
    - The specific security attributes used in making an information flow enforcement decision.
    - Some specific subsets of the information that has flowed based upon policy goals (e.g. auditing of downgraded material).
16. Audit events for FDP_IFF.3 (See 5.1.4):
    - Decisions to permit requested information flows.
    - All decisions on requests for information flow.
    - The use of identified illicit information flow channels.
    - The specific security attributes used in making an information flow enforcement decision.
    - Some specific subsets of the information that has flowed based upon policy goals (e.g. auditing of downgraded material).
    - The use of identified illicit information flow channels with estimated maximum capacity exceeding a specified value.
17. Audit events for FDP_ITC.1 (See 5.1.4):
    - Successful import of user data, including any security attributes.
    - All attempts to import user data, including any security attributes.
    - The specification of security attributes for imported user data supplied by an authorised user.
18. Audit events for FDP_ITC.2 (See 5.1.4):
    - Successful import of user data, including any security attributes.
    - All attempts to import user data, including any security attributes.
    - The specification of security attributes for imported user data supplied by an authorised user.
19. Audit events for FDP_ITT.2 (See 5.1.4):
    - Successful transfers of user data, including identification of the protection method used.
    - All attempts to transfer user data, including the protection method used and any errors that occurred.
20. Audit events for FDP_SDI.2 (See 5.1.4):
    - Successful attempts to check the integrity of user data, including an indication of the results of the check.
    - All attempts to check the integrity of user data, including an indication of the results of the check, if performed.
    - The type of integrity error that occurred.
    - The action taken upon detection of an integrity error.
21. Audit events for FDP_UIT.1 (See 5.1.4):
    - The identity of any user or subject using the data exchange mechanisms.
    - The identity of any user or subject attempting to use the user data exchange mechanisms, but who is unauthorised to do so.
    - A reference to the names or other indexing information useful in identifying the user data that was transmitted or received. This could include security attributes associated with the user data.

- Any identified attempts to block transmission of user data.
- The types and/or effects of any detected modifications of transmitted user data.

22. Audit events for FDP_UIT.3 (See 5.1.4):
    - The identity of any user or subject using the data exchange mechanisms.
    - Successful recovery from errors including they type of error that was detected.
    - The identity of any user or subject attempting to use the user data exchange mechanisms, but who is unauthorised to do so.
    - A reference to the names or other indexing information useful in identifying the user data that was transmitted or received. This could include security attributes associated with the user data.
    - Any identified attempts to block transmission of user data.
    - The types and/or effects of any detected modifications of transmitted user data.
23. Audit events for FIA_AFL.1 (See 5.1.5):
    - The reaching of the threshold for the unsuccesful authentication attempts and the actions (e.g. disabling of a terminal) taken and the subsequent, if appropriate, restoration to the normal state (e.g. re-enabling of a terminal).
24. Audit events for FIA_SOS.1 (See 5.1.5):
    - Rejection by the TSF of any tested secret;
    - Rejection or acceptance by the TSF of any tested secret;
    - Identification of any changes to the defined quality metrics.
25. Audit events for FIA_SOS.2 (See 5.1.5):
    - Rejection by the TSF of any tested secret;
    - Rejection or acceptance by the TSF of any tested secret;
    - Identification of any changes to the defined quality metrics.
26. Audit events for FIA_UAU.2 (See 5.1.5):
    - Unsuccessful use of the authentication mechanism;
    - All use of the authentication mechanism.
27. Audit events for FIA_UAU.6 (See 5.1.5):
    - Failure of reauthentication;
    - All reauthentication attempts.
28. Audit events for FIA_UID.2 (See 5.1.5):
    - Unsuccessful use of the user identification mechanism, including the user identity provided;
    - All use of the user identification mechanism, including the user identity provided.
29. Audit events for FIA_USB.1 (See 5.1.5):
    - Unsuccessful binding of user security attributes to a subject (e.g. creation of a subject).
    - Success and failure of binding of user security attributes to a subject (e.g. success and failure to create a subject).
30. Audit events for FMT_MOF.1 (See 5.1.6):
    - All modifications in the behaviour of the functions in the TSF.
31. Audit events for FMT_MSA.1 (See 5.1.6):

   - All modifications of the values of security attributes.

32. Audit events for FMT_MSA.2 (See 5.1.6):
    - All offered and rejected values for a security attribute;
    - All offered and accepted secure values for a security attribute.
33. Audit events for FMT_MSA.3 (See 5.1.6):
    - Modifications of the default setting of permissive or restrictive rules.
    - All modifications of the initial values of security attributes.
34. Audit events for FMT_MTD.1 (See 5.1.6):
    - All modifications to the values of TSF data.
35. Audit events for FMT_MTD.2 (See 5.1.6):
    - All modifications to the limits on TSF data;
    - All modifications in the actions to be taken in case of violation of the limits.
36. Audit events for FMT_MTD.3 (See 5.1.6):
    - All rejected values of TSF data.
37. Audit events for FMT_REV.1 (See 5.1.6):
    - Unsuccessful revocation of security attributes;
    - All attempts to revoke security attributes.
38. Audit events for FMT_SAE.1 (See 5.1.6):
    - Specification of the expiration time for an attribute;
    - Action taken due to attribute expiration.
39. Audit events for FMT_SMR.2 (See 5.1.6):
    - Modifications to the group of users that are part of a role;
    - unsuccessful attempts to use a role due to the given conditions on the roles;
    - Every use of the rights of a role.
40. Audit events for FPT_AMT.1 (See 5.1.7):
    - Execution of the tests of the underlying machine and the results of the tests.
41. Audit events for FPT_FLS.1 (See 5.1.7):
    - Failure of the TSF.
42. Audit events for FPT_ITI.1 (See 5.1.7):
    - The detection of modification of transmitted TSF data.
    - The action taken upon detection of modification of transmitted TSF data.
43. Audit events for FPT_RCV.1 (See 5.1.7):
    - The fact that a failure or service discontinuity occurred;
    - resumption of the regular operation;
    - Type of failure or service discontinuity.
44. Audit events for FPT_RCV.4 (See 5.1.7):
    - If possible, the impossibility to return to a secure state after failure of a security function;
    - If possible, the detection of a failure of a security function.
45. Audit events for FPT_SSP.2 (See 5.1.7):
    - Failure to receive an acknowledgement when expected.
46. Audit events for FPT_STM.1 (See 5.1.7):
    - Changes to the time;

- Providing a timestamp.

47. Audit events for FPT_TDC.1 (See 5.1.7):
    - Successful use of TSF data consistency mechanisms.
    - Use of the TSF data consistency mechanisms.
    - Identification of which TSF data have been interpreted.
    - Detection of modified TSF data.
48. Audit events for FPT_TRC.1 (See 5.1.7):
    - Restoring consistency upon reconnection.
    - Detected inconsistency between TSF data.
49. Audit events for FPT_TST.1 (See 5.1.7):
    - Execution of the TSF self tests and the results of the tests.
50. Audit events for FRU_PRS.2 (See 5.1.8):
    - Rejection of operation based on the use of priority within an allocation.
    - All attempted uses of the allocation function which involves the priority of the service functions.
51. Audit events for FRU_RSA.2 (See 5.1.8):
    - Rejection of allocation operation due to resource limits.
    - All attempted uses of the resource allocation functions for resources that are under control of the TSF.
52. Audit events for FTA_MCS.2 (See 5.1.9):
    - Rejection of a new session based on the limitation of multiple concurrent sessions.
    - Capture of the number of currently concurrent user sessions and the user security attribute(s).
53. Audit events for FTA_SSL.1 (See 5.1.9):
    - Locking of an interactive session by the session locking mechanism.
    - Successful unlocking of an interactive session.
    - Any attempts at unlocking an interactive session.
54. Audit events for FTA_SSL.2 (See 5.1.9):
    - Locking of an interactive session by the session locking mechanism.
    - Successful unlocking of an interactive session.
    - Any attempts at unlocking an interactive session.
55. Audit events for FTA_SSL.3 (See 5.1.9):
    - Termination of an interactive session by the session locking mechanism.
56. Audit events for FTA_TSE.1 (See 5.1.9):
    - Denial of a session establishment due to the session establishment mechanism.
    - All attempts at establishment of a user session.
    - Capture of the value of the selected access parameters (e.g. location of access, time of access).
57. Audit events for FTP_ITC.1 (See 5.1.10):
    - Failure of the trusted channel functions.
    - Identification of the initiator and target of failed trusted channel functions.
    - All attempted uses of the trusted channel functions.
    - Identification of the initiator and target of all trusted channel functions.

58. Audit events for FTP_TRP.1 (See 5.1.10):
    - Failures of the trusted path functions.
    - Identification of the user associated with all trusted path failures, if available.
    - All attempted uses of the trusted path functions.
    - Identification of the user associated with all trusted path invocations, if available.

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

- Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event; and
- For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [ST ASSIGNMENT: **(other audit relevant information)**]

FAU_GEN.2 User identity association

User identity association, the TSF shall associate auditable events to individual user identities.

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

FAU_SAR - Security audit review

This family defines the requirements for audit tools that should be available to authorised users to assist in the review of audit data.

FAU_SAR.1 Audit review

Audit review provides the capability to read information from the audit records.

FAU_SAR.1.1 The TSF shall provide [ST ASSIGNMENT: **(authorised users)**] with the capability to read [ST ASSIGNMENT: **(list of audit information)**] from the audit records.

FAU_SAR.1.2 The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

FAU_SAR.3 Selectable audit review

Selectable audit review requires audit review tools to select the audit data to be reviewed based on criteria.

FAU_SAR.3.1 The TSF shall provide the ability to perform [ST SELECTION: **(searches, sorting, ordering)**] of audit data based on [ST ASSIGNMENT: **(criteria with logical relations)**].

FAU_SEL - Security audit event selection

This family defines requirements to select the events to be audited during TOE operation. It defines requirements to include or exclude events from the set of auditable events.

FAU_SEL.1 Selective audit

Selective audit, requires the ability to include or exclude events from the set of audited events based upon attributes to be specified by the PP/ST author.

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

- - [ST SELECTION: **(object identity, user identity, subject identity, host identity, event type)**]
- - [ST ASSIGNMENT: **(list of additional attributes that audit selectivity is based upon)**].

FAU_STG - Security audit event storage

This family defines the requirements for the TSF to be able to create and maintain a secure audit trail.

FAU_STG.2 Guarantees of audit data availability

Guarantees of audit data availability specifies the guarantees that the TSF maintains over the audit data given the occurrence of an undesired condition.

FAU_STG.2.1 The TSF shall protect the stored audit records from unauthorised deletion.

FAU_STG.2.2 The TSF shall be able to **detect** modifications to the audit records.

FAU_STG.2.3 The TSF shall ensure that [ST ASSIGNMENT: **(metric for saving audit records)**] audit records will be maintained when the following conditions occur: [ST SELECTION: **(audit storage exhaustion, failure, attack)**].

FAU_STG.4 Prevention of audit data loss

Prevention of audit data loss specifies actions in case the audit trail is full.

FAU_STG.4.1 The TSF shall **prevent auditable events, except those taken by the authorized user with special rights** and extbfoverwrite the oldest stored audit records for these users, creating a circular log area if the audit trail is full.

5.1.2 FCO - Communication

This class provides two families specifically concerned with assuring the identity of a party participating in a data exchange. These families are related to assuring the identity of the originator of transmitted information (proof of origin) and assuring the identity of the recipient of transmitted information (proof of receipt). These families ensure that an originator cannot deny having sent the message, nor can the recipient deny having received it.

FCO_NRO - Non-repudiation of origin

Non-repudiation of origin ensures that the originator of information cannot successfully deny having sent the information. This family requires that the TSF provide a method to ensure that a subject that receives information during a data exchange is provided with evidence of the origin of the information. This evidence can then be verified by either this subject or other subjects.

FCO_NRO.2 Enforced proof of origin

Enforced proof of origin requires that the TSF always generate evidence of origin for transmitted information.

FCO_NRO.2.1 The TSF shall enforce the generation of evidence of origin for transmitted **certificates, root certificates, scripts and configuration files** at all times.

FCO_NRO.2.2 The TSF shall be able to relate the [ST ASSIGNMENT: **(list of attributes)**] of the originator of the information, and the [ST ASSIGNMENT: **(list of information fields)**] of the information to which the evidence applies.

FCO_NRO.2.3 The TSF shall provide a capability to verify the evidence of origin of information to [ST SELECTION: **(originator, recipient, [ST ASSIGNMENT: (list of third parties)])**] given [ST ASSIGNMENT: **(limitations on the evidence of origin)**].

5.1.3 FCS - Cryptographic support

The TSF may employ cryptographic functionality to help satisfy several high-level security objectives. These include (but are not limited to): identification and authentication, non-repudiation, trusted path, trusted channel and data separation. This class is used when the TOE implements cryptographic functions, the implementation of which could be in hardware, firmware and/or software.

FCS_CKM - Cryptographic key management

Cryptographic keys must be managed throughout their life cycle. This family is intended to support that lifecycle and consequently defines requirements for the following activities: cryptographic key generation, cryptographic key distribution, cryptographic key access and cryptographic key destruction. This family should be included whenever there are functional requirements for the management of cryptographic keys.

FCS_CKM.1 Cryptographic key generation

Cryptographic key generation requires cryptographic keys to be generated in accordance with a specified algorithm and key sizes which can be based on an assigned standard.

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [ST ASSIGNMENT: **(cryptographic key generation algorithm)**] and specified cryptographic key sizes [ST ASSIGNMENT: **(cryptographic key sizes)**] that meet the following: [ST ASSIGNMENT: **(list of standards)**].

These algorithms, key sizes and standards are referred herein as "Approved".

Secret keys, private keys, and CSPs shall be protected within the TSF from unauthorized disclosure, modification, and substitution. Public keys shall be protected within the TSF against unauthorized modification and substitution.

Random Number Generators (RNGs)

A TSF may employ random number generators (RNGs). If a TSF employs Approved or non-Approved RNGs in an Approved mode of operation, the data output from the RNG shall pass the continuous random number generator test as specified in 5.1.7 . The data output from an Approved RNG shall pass all statistical tests for randomness as specified in 5.1.7. Approved deterministic RNGs shall be subject to the cryptographic algorithm test in 5.1.7.

An Approved RNG shall be used for the generation of cryptographic keys used by an Approved security function. The output from a non-Approved RNG may be used 1) as input (e.g., seed, and seed key) to an Approved deterministic RNG or 2) to generate initialization vectors (IVs) for Approved security function(s). The seed and seed key shall not have the same value.

Key Generation

A TSF may generate cryptographic keys internally. Cryptographic keys generated by the TSF for use by an Approved algorithm or security function shall be generated using an Approved key generation method. If an Approved key generation method requires input from a RNG, then an Approved RNG that meets the stated requirements shall be used.

Compromising the security of the key generation method (e.g., guessing the seed value to initialize the deterministic RNG) shall require as least as many operations as determining the value of the generated key.

If a seed key is entered during the key generation process, entry of the key shall meet the key entry requirements specified above. If intermediate key generation values are output from the TSF, the values shall be output either 1) in encrypted form or 2) under split knowledge procedures.

FCS_CKM.2 Cryptographic key distribution

Cryptographic key distribution requires cryptographic keys to be distributed in accordance with a specified distribution method which can be based on an assigned standard.

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method [ST ASSIGNMENT: **(cryptographic key distribution method)**] that meets the following: [ST ASSIGNMENT: **(list of standards)**].

These distribution methods and standards are referred herein as "Approved".

Key Establishment

Key establishment may be performed by automated methods (e.g., use of a public key algorithm), manual methods (use of a manually-transported key loading device), or a combination of automated and manual methods. If key establishment methods are employed by a TSF, only Approved key establishment methods shall be used. If, in lieu of an Approved key establishment method, a radio communications TSF implements Over-The-Air-Rekeying, it shall be implemented as specified in the citetia.

Compromising the security of the key establishment method (e.g., compromising the security of the algorithm used for key establishment) shall require at least as many operations as determining the value of the cryptographic key being transported or agreed upon.

If a key transport method is used, the cryptographic key being transported shall meet the key following entry/output requirements.

If a key agreement method is used (e.g., a cryptographic key is derived from shared intermediate values), the shared values are not required to meet these key entry/output requirements.

Key Entry and Output

Cryptographic keys may be entered into or output from a TSF. If cryptographic keys are entered into or output from a TSF, the entry or output of keys shall be performed using either manual (e.g., via a keyboard) or electronic methods (e.g., smart cards/tokens, PC cards, or other electronic key loading devices).

A seed key, if entered during key generation, shall be entered in the same manner as cryptographic keys.

All encrypted secret and private keys, entered into or output from a TSF and used in an Approved mode of operation, shall be encrypted using an Approved algorithm. Public keys may be entered into or output from a TSF in plaintext form. A TSF shall associate a key (secret, private, or public) entered into or output from the module with the correct entity (i.e., person, group, or process) to which the key is assigned.

Manually-entered cryptographic keys (keys entered using manual methods) shall be verified during entry into a TSF for accuracy using the manual key entry test specified in 5.1.7 During key entry, the manually entered values may be temporarily displayed to allow visual verification and to improve accuracy. The plaintext values of the cryptographic keys or key components shall not be displayed.

Secret and private keys established using automated methods shall be entered into and output from a TSF in encrypted form.

Secret and private keys established using manual methods shall be entered into or output from a TSF either (1) in encrypted form or (2) using split knowledge procedures (i.e., as two or more plaintext cryptographic key components).

If split knowledge procedures are used:

- the TSF shall separately authenticate the operator entering or outputting each key component,
- plaintext cryptographic key components shall be directly entered into or output from the TSF (e.g., via a trusted path or directly attached cable) without traveling through any enclosing or intervening systems where the key components may inadvertently be stored, combined, or otherwise processed,
- at least two key components shall be required to reconstruct the original cryptographic key.

FCS_CKM.3 Cryptographic key access

Cryptographic key access requires access to cryptographic keys to be performed in accordance with a specified access method which can be based on an assigned standard.

FCS_CKM.3.1 The TSF shall perform [ST ASSIGNMENT: **(type of cryptographic key access)**] in accordance with a specified cryptographic key access method [ST ASSIGNMENT: **(cryptographic key access method)**] that meets the following: [ST ASSIGNMENT: **(list of standards)**].

Key Storage

Cryptographic keys stored within a TSF shall be stored either in plaintext form or encrypted form. Plaintext secret and private keys shall not be accessible from outside the TSF to unauthorized operators.

A TSF shall associate a cryptographic key (secret, private, or public) stored within the module with the correct entity (e.g., person, group, or process) to which the key is assigned.

FCS_CKM.4 Cryptographic key destruction

Cryptographic key destruction requires cryptographic keys to be destroyed in accordance with a specified destruction method which can be based on an assigned standard.

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [ST ASSIGNMENT: **(cryptographic key destruction method)**] that meets the following: [ST ASSIGNMENT: **(list of standards)**].

Key Zeroization

A TSF shall provide methods to zeroize all plaintext secret and private cryptographic keys and CSPs within the module. Zeroization of encrypted cryptographic keys and CSPs or keys otherwise physically or logically protected within an additional embedded module that complies with this PP is not required.

FCS_COP - Cryptographic operation

In order for a cryptographic operation to function correctly, the operation must be performed in accordance with a specified algorithm and with a cryptographic key of a specified size. This

family should be included whenever there are requirements for cryptographic operations to be performed.
Typical cryptographic operations include data encryption and/or decryption, digital signature generation and/or verification, cryptographic checksum generation for integrity and/or verification of checksum, secure hash (message digest), cryptographic key encryption and/or decryption, and cryptographic key agreement.

FCS_COP.1 Cryptographic operation

Cryptographic operation requires a cryptographic operation to be performed in accordance with a specified algorithm and with a cryptographic key of specified sizes. The specified algorithm and cryptographic key sizes can be based on an assigned standard.

FCS_COP.1.1 The TSF shall perform [ST ASSIGNMENT: **(list of cryptographic operations)**] in accordance with a specified cryptographic algorithm [ST ASSIGNMENT: **(cryptographic algorithm)**] and cryptographic key sizes [ST ASSIGNMENT: **(cryptographic key sizes)**] that meet the following: [ST ASSIGNMENT: **(list of standards)**].

5.1.4 FDP - User data protection

This class contains families specifying requirements for TOE security functions and TOE security function policies related to protecting user data. FDP is split into four groups of families (listed below) that address user data within a TOE, during import, export, and storage as well as security attributes directly related to user data.

FDP_ACC - Access control policy

This family identifies the access control SFPs (by name) and defines the scope of control of the policies that form the identified access control portion of the TSP. This scope of control is characterised by three sets: the subjects under control of the policy, the objects under control of the policy, and the operations among controlled subjects and controlled objects that are covered by the policy. The criteria allows multiple policies to exist, each having a unique name. This is accomplished by iterating components from this family once for each named access control policy. The rules that define the functionality of an access control SFP will be defined by other families such as FDP_ACF and FDP_SDI. The names of the access control SFPs identified here in FDP_ACC are meant to be used throughout the remainder of the functional components that have an operation that calls for an assignment or selection of an "access control SFP."

FDP_ACC.2 Complete access control

Complete access control requires that each identified access control SFP cover all operations on subjects and objects covered by that SFP. It further requires that all objects and operations with the TSC are covered by at least one identified access control SFP.

FDP_ACC.2.1 The TSF shall enforce the **Role Based Access Control, as defined in [Nat00] with static separation of duties** on **all TOE subjects and objects** and all operations among subjects and objects covered by the SFP.

A TSF may permit an authenticated operator to perform all of the services allowed within an authorized role, or may require separate authentication for each service or for different sets of services. When a TSF is powered off and subsequently powered on, the results of previous authentications shall not be retained and the TSF shall require the operator to be re-authenticated.

Various types of authentication data may be required by a TSF to implement the supported authentication mechanisms, including (but not limited to) the knowledge or possession of a password, PIN, cryptographic key, or equivalent; possession of a physical key, token, or equivalent; or verification of personal characteristics (e.g., biometrics). Authentication data within a TSF shall be protected against unauthorized disclosure, modification, and substitution.

The initialization of authentication mechanisms may warrant special treatment. If a TSF does not contain the authentication data required to authenticate the operator for the first time the module is accessed, then other authorized methods (e.g., procedural controls or use of factory-set or default authentication data) shall be used to control access to the module and initialize the authentication mechanisms.

FDP_ACC.2.2 The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

FDP_ACF - Access control functions

This family describes the rules for the specific functions that can implement an access control policy named in FDP_ACC. FDP_ACC specifies the scope of control of the policy.

FDP_ACF.1 Security attribute based access control

Security attribute based access control allows the TSF to enforce access based upon security attributes and named groups of attributes. Furthermore, the TSF may have the ability to explicitly authorise or deny access to an object based upon security attributes.

FDP_ACF.1.1 The TSF shall enforce the [ST ASSIGNMENT: **(access control SFP)**] to objects based on [ST ASSIGNMENT: **(security attributes, named groups of security attributes)**].

FDP_ACF.1.2 The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [ST ASSIGNMENT: **(rules governing access among controlled subjects and controlled objects using controlled operations on controlled objects)**].

FDP_ACF.1.3 The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: [ST ASSIGNMENT: **(rules, based on security attributes, that explicitly authorise access of subjects to objects)**].

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to objects based on the [ST ASSIGNMENT: **(rules, based on security attributes, that explicitly deny access of subjects to objects)**].

FDP_DAU - Data authentication

Data authentication permits an entity to accept responsibility for the authenticity of information (e.g., by digitally signing it). This family provides a method of providing a guarantee of the validity of a specific unit of data that can be subsequently used to verify that the information content has not been forged or fraudulently modified. In contrast to Class FAU, this family is intended to be applied to "static" data rather than data that is being transferred.

FDP_DAU.2 Data authentication with identity of guarantor

Data Authentication with Identity of Guarantor additionally requires that the TSF is capable of establishing the identity of the subject who provided the guarantee of authenticity.

FDP_DAU.2.1 The TSF shall provide a capability to generate evidence that can be used as a guarantee of the validity of [ST ASSIGNMENT: **(list of objects or information types)**].

FDP_DAU.2.2 The TSF shall provide [ST ASSIGNMENT: **(list of subjects)**] with the ability to verify evidence of the validity of the indicated information and the identity of the user that generated the evidence.

FDP_ETC - Export to outside TSF control

This family defines functions for exporting user data from the TOE such that its security attributes and protection either can be explicitly preserved or can be ignored once it has been exported. It is concerned with limitations on export and with the association of security attributes with the exported user data.

FDP_ETC.1 Export of user data without security attributes

Export of user data without security attributes requires that the TSF enforce the appropriate SFPs when exporting user data outside the TSF. User data that is exported by this function is exported without its associated security attributes.

FDP_ETC.1.1 The TSF shall enforce the [ST ASSIGNMENT: **(access control SFP(s) and/or information flow control SFP(s))**] when exporting user data, controlled under the SFP(s), outside of the TSC.

FDP_ETC.1.2 The TSF shall export the user data without the user data's associated security attributes.

FDP_ETC.2 Export of user data with security attributes

Export of user data with security attributes requires that the TSF enforce the appropriate SFPs using a function that accurately and unambiguously associates security attributes with the user data that is exported.

FDP_ETC.2.1 The TSF shall enforce the [ST ASSIGNMENT: **(access control SFP(s) and/or information flow control SFP(s))**] when exporting user data, controlled under the SFP(s), outside of the TSC.

FDP_ETC.2.2 The TSF shall export the user data with the user data's associated security attributes.

FDP_ETC.2.3 The TSF shall ensure that the security attributes, when exported outside the TSC, are unambiguously associated with the exported user data.

FDP_ETC.2.4 The TSF shall enforce the following rules when user data is exported from the TSC: [ST ASSIGNMENT: **(additional exportation control rules)**].

FDP_IFC - Information flow control policy

This family identifies the information flow control SFPs (by name) and defines the scope of control of the policies that form the identified information flow control portion of the TSP. This scope of control is characterised by three sets: the subjects under control of the policy, the information under control of the policy, and operations which cause controlled information to flow to and from controlled subjects covered by the policy. The criteria allows multiple policies to exist, each having a unique name. This is accomplished by iterating components from this family once for each named information flow control policy. The rules that define the functionality of an information flow control SFP will be defined by other families such as FDP_IFF and FDP_SDI. The names of the information flow control SFPs identified here in FDP_IFC are meant to be used throughout the remainder of the functional components that have an operation that calls for an assignment or selection of an "information flow control SFP."

FDP_IFC.1 Subset information flow control

Subset information flow control requires that each identified information flow control SFPs be in place for a subset of the possible operations on a subset of information flows in the TOE.

FDP_IFC.1.1 The TSF shall enforce **the following information control flow model and policy** on **all subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP**:

Iff model

Two levels of nondisclosure are defined, L_k, secret level, where private keys are known, and L_p, public level, whose objects are the results of applying cryptographic functions to those keys at L_k.

The system is said to be secure if the contents of L_k are only known to the defined set of allowed subjects. No restriction is applied to L_p, where universal access is granted to its content.

Initially, the system is as secure as it will ever be. On using the keys at L_k, L_p is populated with the results of every usage, and each one is a small leakage of the secrecy of the applied key. System security degrades over L_k usage, and the speed of this degradation is proportional to the weakness of the cryptographic function and used keys.

System insecurity can be modeled as the certainty that any of its secrets at L_k can be publicly known:

$$SI = \max_{k=0}^{n} \sum_{f=0}^{m} \frac{CARD(L_{p_{f(k)}})}{STRENGTH(f)}$$

where

SI is the system insecurity value.

$CARD(L_{p_{f(k)}})$ is the cardinality of function f applications on key k.

$STRENGTH(f)$ is the current strength of function f to cryptanalysis [1]

The bottom line is that L_k and L_p do not follow general iff models where information can only flow to upper levels. In a PKI case, the secret is slowly transmitted to the lower or public level, and this is a tolerated leakage. The above function can be used to determine the system secure life time.

Iff policy

1. No complete disclosure of the private keys at L_k shall be allowed by the system.
2. L_k secrecy leakages shall only be allowed by the application of cryptographic functions.

This allows to verify the system implementation, in that no system operation or information flow path violates the first rule, and that the only disclosure channels are those approved cryptographic functions.

Note that the intended system security level is directly affected by the strength of the cryptographic algorithms, key sizes and standards that must be instantiated at the TOE ST as defined at FCS (See 5.1.3).

FDP_IFF - Information flow control functions

This family descibes the rules for the specific functions that can implement the information flow control SFPs named in FDP_IFC, which also specifies the scope of control of the policy. It consists of two kinds of requirements: one addressing the common information flow function

[1] See [Bor44] for a detailed definition of future cryptanalysis techniques and methods.

issues, and a second addressing illicit information flows (i.e. covert channels). This division arises because the issues concerning illicit information flows are, in some sense, orthogonal to the rest of an information flow control SFP. By their nature they circumvent the information flow control SFP resulting in a violation of the policy. As such, they require special functions to either limit or prevent their occurrence.

FDP_IFF.1 Simple security attributes

Simple security attributes requires security attributes on information, and on subjects that cause that information to flow and on subjects that act as recipients of that information. It specifies the rules that must be enforced by the function, and describes how security attributes are derived by the function.

FDP_IFF.1.1 The TSF shall enforce the [ST ASSIGNMENT: **(information flow control SFP)**] based on the following types of subject and information security attributes: [ST ASSIGNMENT: **(the minimum number and type of security attributes)**].

FDP_IFF.1.2 The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: [ST ASSIGNMENT: **(for each operation, the security attribute-based relationship that must hold between subject and information security attributes)**].

FDP_IFF.1.3 The TSF shall enforce the [ST ASSIGNMENT: **(additional information flow control SFP rules)**].

FDP_IFF.1.4 The TSF shall provide the following [ST ASSIGNMENT: **(list of additional SFP capabilities)**].

FDP_IFF.1.5 The TSF shall explicitly authorise an information flow based on the following rules: [ST ASSIGNMENT: **(rules, based on security attributes, that explicitly authorise information flows)**].

FDP_IFF.1.6 The TSF shall explicitly deny an information flow based on the following rules: [ST ASSIGNMENT: **(rules, based on security attributes, that explicitly deny information flows)**].

FDP_IFF.3 Limited illicit information flows

Limited illicit information flows requires the SFP to cover illicit information flows, but not necessarily eliminate them.

FDP_IFF.3.1 The TSF shall enforce the [ST ASSIGNMENT: **(information flow control SFP)**] to limit the capacity of [ST ASSIGNMENT: **(types of illicit information flows)**] to a [ST ASSIGNMENT: **(maximum capacity)**].

FDP_ITC - Import from outside TSF control

This family defines the mechanisms for introduction of user data into the TOE such that it has appropriate security attributes and is appropriately protected. It is concerned with limitations on importation, determination of desired security attributes, and interpretation of security attributes associated with the user data.

FDP_ITC.1 Import of user data without security attributes

Import of user data without security attributes requires that the security attributes correctly represent the user data and are supplied separately from the object.

FDP_ITC.1.1 The TSF shall enforce the [ST ASSIGNMENT: **(access control SFP and/or information flow control SFP)**] when importing user data, controlled under the SFP, from outside of the TSC.

FDP_ITC.1.2 The TSF shall ignore any security attributes associated with the user data when imported from outside the TSC.

FDP_ITC.1.3 The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: [ST ASSIGNMENT: **(additional importation control rules)**].

FDP_ITC.2 Import of user data with security attributes

Import of user data with security attributes requires that security attributes correctly represent the user data and are accurately and unambiguously associated with the user data imported from outside the TSC.

FDP_ITC.2.1 The TSF shall enforce the [ST ASSIGNMENT: **(access control SFP and/or information flow control SFP)**] when importing user data, controlled under the SFP, from outside of the TSC.

FDP_ITC.2.2 The TSF shall use the security attributes associated with the imported user data.

FDP_ITC.2.3 The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

FDP_ITC.2.4 The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

FDP_ITC.2.5 The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: [ST ASSIGNMENT: **(additional importation control rules)**].

FDP_ITT - Internal TOE transfer

This family provides requirements that address protection of user data when it is transferred between parts of a TOE across an internal channel. This may be contrasted with the FDP_UCT and FDP_UIT families, which provide protection for user data when it is transferred between distinct TSFs across an external channel, and FDP_ETC and FDP_ITC, which address transfer of data to or from outside the TSF's control.

FDP_ITT.2 Transmission separation by attribute

Transmission separation by attribute requires separation of data based on the value of SFP-relevant attributes in addition to the first component.

FDP_ITT.2.1 The TSF shall enforce the [ST ASSIGNMENT: **(access control SFP(s) and/or information flow control SFP(s))**] to prevent the [ST SELECTION: **(disclosure, modification, loss of use)**] of user data when it is transmitted between physically-separated parts of the TOE.

FDP_ITT.2.2 The TSF shall separate data controlled by the SFP(s) when transmitted between physically-separated parts of the TOE, based on the values of the following: [ST ASSIGNMENT: **(security attributes that require separation)**].

FDP_RIP - Residual information protection

This family addresses the need to ensure that deleted information is no longer accessible, and that newly created objects do not contain information that should not be accessible. This family requires protection for information that has been logically deleted or released, but may still be present within the TOE.

FDP_RIP.2 Full residual information protection

Full residual information protection requires that the TSF ensure that any residual information content of any resources is unavailable to all objects upon the resource's allocation or deallocation.

FDP_RIP.2.1 The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** all objects.

FDP_SDI - Stored data integrity

This family provides requirements that address protection of user data while it is stored within the TSC. Integrity errors may affect user data stored in memory, or in a storage device. This family differs from FDP_ITT Internal TOE transfer which protects the user data from integrity errors while being transferred within the TOE.

FDP_SDI.2 Stored data integrity monitoring and action

Stored data integrity monitoring and action adds the additional capability to the first component by allowing for actions to be taken as a result of an error detection.

FDP_SDI.2.1 The TSF shall monitor user data stored within the TSC for [ST ASSIGNMENT: **(integrity errors)**] on all objects, based on the following attributes: [ST ASSIGNMENT: **(user data attributes)**].

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall [ST ASSIGNMENT: **(action to be taken)**].

FDP_UIT - Inter-TSF user data integrity transfer protection

This family defines the requirements for providing integrity for user data in transit between the TSF and another trusted IT product and recovering from detectable errors. At a minimum, this family monitors the integrity of user data for modifications. Furthermore, this family supports different ways of correcting detected integrity errors.

FDP_UIT.1 Data exchange integrity

Data exchange integrity addresses detection of modifications, deletions, insertions, and replay errors of the user data transmitted.

FDP_UIT.1.1 The TSF shall enforce the [ST ASSIGNMENT: **(access control SFP(s) and/or information flow control SFP(s))**] to be able to [ST SELECTION: **(transmit, receive)**] user data in a manner protected from [ST SELECTION: **(modification, deletion, insertion, replay)**] errors.

FDP_UIT.1.2 The TSF shall be able to determine on receipt of user data, whether [ST SELECTION: **(modification, deletion, insertion, replay)**] has occurred.

FDP_UIT.3 Destination data exchange recovery

Destination data exchange recovery addresses recovery of the original user data by the receiving TSF on its own without any help from the source trusted IT product.

FDP_UIT.3.1 The TSF shall enforce the [ST ASSIGNMENT: **(access control SFP(s) and/or information flow control SFP(s))**] to be able to recover from [ST ASSIGNMENT: **(list of recoverable errors)**] without any help from the source trusted IT product.

5.1.5 FIA - Identification and authentication

Identification and Authentication is required to ensure that users are associated with the proper security attributes (e.g. identity, groups, roles, security or integrity levels).
The unambiguous identification of authorised users and the correct association of security attributes with users and subjects is critical to the enforcement of the intended security policies. The families in this class deal with determining and verifying the identity of users, determining their authority to interact with the TOE, and with the correct association of security attributes for each authorised user. Other classes of requirements (e.g. User Data Protection, Security Audit) are dependent upon correct identification and authentication of users in order to be effective.

FIA_AFL - Authentication failures

This family contains requirements for defining values for some number of unsuccessful authentication attempts and TSF actions in cases of authentication attempt failures. Parameters include, but are not limited to, the number of failed authentication attempts and time thresholds.

FIA_AFL.1 Authentication failure handling

Requires that the TSF be able to terminate the session establishment process after a specified number of unsuccessful user authentication attempts. It also requires that, after termination of the session establishment process, the TSF be able to disable the user account or the point of entry (e.g. workstation) from which the attempts were made until an administrator-defined condition occurs.

FIA_AFL.1.1 The TSF shall detect when [ST ASSIGNMENT: **(number)**] unsuccessful authentication attempts occur related to [ST ASSIGNMENT: **(list of authentication events)**].

FIA_AFL.1.2 When the defined number of unsuccessful authentication attempts has been met or surpassed, the TSF shall [ST ASSIGNMENT: **(list of actions)**].

FIA_ATD - User attribute definition

All authorised users may have a set of security attributes, other than the user's identity, that is used to enforce the TSP. This family defines the requirements for associating user security attributes with users as needed to support the TSP.

FIA_ATD.1 User attribute definition

User attribute definition, allows user security attributes for each user to be maintained individually.

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users: [ST ASSIGNMENT: **(list of security attributes)**].

FIA_SOS - Specification of secrets

This family defines requirements for mechanisms that enforce defined quality metrics on provided secrets and generate secrets to satisfy the defined metric.

FIA_SOS.1 Verification of secrets

Verification of secrets requires the TSF to verify that secrets meet defined quality metrics.

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

- For each attempt to use the authentication mechanism, the probability shall be less than one in 1,000,000 that a random attempt will succeed or a false acceptance will occur (e.g., guessing a password or PIN, false acceptance error rate of a biometric device, or some combination of authentication methods).

- For multiple attempts to use the authentication mechanism during a one-minute period, the probability shall be less than one in 100,000 that a random attempt will succeed or a false acceptance will occur.
- Feedback of authentication data to an operator shall be obscured during authentication (e.g., no visible display of characters when entering a password).
- Feedback provided to an operator during an attempted authentication shall not weaken the strength of the authentication mechanism.

FIA_SOS.2 TSF Generation of secrets

Generation of secrets requires the TSF to be able to generate secrets that meet defined quality metrics.

FIA_SOS.2.1 The TSF shall provide a mechanism to generate secrets that meet [ST ASSIGNMENT: **(a defined quality metric)**].

FIA_SOS.2.2 The TSF shall be able to enforce the use of TSF generated secrets for [ST ASSIGNMENT: **(list of TSF functions)**].

FIA_UAU - User authentication

This family defines the types of user authentication mechanisms supported by the TSF. This family also defines the required attributes on which the user authentication mechanisms must be based.

FIA_UAU.2 User authentication before any action

User authentication before any action, requires that users authenticate themselves before any action will be allowed by the TSF.

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.

FIA_UAU.6 Re-authenticating

Re-authenticating, requires the ability to specify events for which the user needs to be re-authenticated.

FIA_UAU.6.1 The TSF shall re-authenticate the user under the conditions [ST ASSIGNMENT: **(list of conditions under which re-authentication is required)**].

FIA_UAU.7 Protected authentication feedback

Protected authentication feedback, require that only limited feedback information is provided to the user during the authentication.

FIA_UAU.7.1 The TSF shall provide only [ST ASSIGNMENT: **(list of feedback)**] to the user while the authentication is in progress.

FIA_UID - User identification

This family defines the conditions under which users shall be required to identify themselves before performing any other actions that are to be mediated by the TSF and which require user identification.

FIA_UID.2 User identification before any action

User identification before any action, require that users identify themselves before any action will be allowed by the TSF.

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 - User-subject binding

An authenticated user, in order to use the TOE, typically activates a subject. The user's security attributes are associated (totally or partially) with this subject. This family defines requirements to create and maintain the association of the user's security attributes to a subject acting on the user's behalf.

FIA_USB.1 User-subject binding

User-subject binding requires the maintenance of an association between the user's security attributes and a subject acting on the user's behalf.

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

5.1.6 FMT - Security management

This class is intended to specify the management of several aspects of the TSF: security attributes, TSF data and functions. The different management roles and their interaction, such as separation of capability, can be specified.

FMT_MOF - Management of functions in TSF

This family allows authorised users control over the management of functions in the TSF. Examples of functions in the TSF include the audit functions and the multiple authentication functions.

FMT_MOF.1 Management of security functions behaviour

Management of security functions behaviour allows the authorised users (roles) to manage the behaviour of functions in the TSF that use rules or have specified conditions that may be manageable.

FMT_MOF.1.1 The TSF shall restrict the ability to [ST SELECTION: **(determine the behavior of, disable, enable, modify the behavior of)**] the functions

1. Management actions due to FAU_SAR.1 (See 5.1.1):
   - Maintenance (deletion, modification, addition) of the group of users with read access right to the audit records.
2. Management actions due to FAU_SEL.1 (See 5.1.1):
   - Maintenance of the rights to view/modify the audit events.
3. Management actions due to FAU_STG.2 (See 5.1.1):
   - Maintenance of the parameters that control the audit storage capability.
4. Management actions due to FAU_STG.4 (See 5.1.1):
   - Maintenance (deletion, modification, addition) of actions to be taken in case of audit storage failure.
5. Management actions due to FCO_NRO.2 (See 5.1.2):
   - The management of changes to information types, fields, originator attributes and recipients of evidence.
6. Management actions due to FCS_CKM.1 (See 5.1.3):
   - The management of changes to cryptographic key attributes. Examples of key attributes include user, key type (e.g. public, private, secret), validity period, and use (e.g. digital signature, key encryption, key agreement, data encryption).
7. Management actions due to FCS_CKM.2 (See 5.1.3):
   - The management of changes to cryptographic key attributes. Examples of key attributes include user, key type (e.g. public, private, secret), validity period, and use (e.g. digital signature, key encryption, key agreement, data encryption).

8. Management actions due to FCS_CKM.3 (See 5.1.3):
   - The management of changes to cryptographic key attributes. Examples of key attributes include user, key type (e.g. public, private, secret), validity period, and use (e.g. digital signature, key encryption, key agreement, data encryption).
9. Management actions due to FCS_CKM.4 (See 5.1.3):
   - The management of changes to cryptographic key attributes. Examples of key attributes include user, key type (e.g. public, private, secret), validity period, and use (e.g. digital signature, key encryption, key agreement, data encryption).
10. Management actions due to FDP_ACF.1 (See 5.1.4):
    - Managing the attributes used to make explicit access or denial based decisions.
11. Management actions due to FDP_DAU.2 (See 5.1.4):
    - The assignment or modification of the objects for which data authentication may apply could be configurable in the system.
12. Management actions due to FDP_ETC.2 (See 5.1.4):
    - The additional exportation control rules could be configurable by a user in a defined role.
13. Management actions due to FDP_IFF.1 (See 5.1.4):
    - Managing the attributes used to make explicit access based decisions.
14. Management actions due to FDP_ITC.1 (See 5.1.4):
    - The modification of the additional control rules used for import.
15. Management actions due to FDP_ITC.2 (See 5.1.4):
    - The modification of the additional control rules used for import.
16. Management actions due to FDP_ITT.2 (See 5.1.4):
    - If the TSF provides multiple methods to protect user data during transmission between physically separated parts of the TOE, the TSF could provide a pre-defined role with the ability to select the method that will be used.
17. Management actions due to FDP_RIP.2 (See 5.1.4):
    - The choice of when to perform residual information protection (i.e. upon allocation or deallocation) could be made configurable within the TOE.
18. Management actions due to FDP_SDI.2 (See 5.1.4):
    - The actions to be taken upon the detection of an integrity error could be configurable.
19. Management actions due to FIA_AFL.1 (See 5.1.5):
    - Management of the threshold for unsuccessful authentication attempts;
    - management of actions to be taken in the event of an authentication failure.
20. Management actions due to FIA_ATD.1 (See 5.1.5):
    - If so indicated in the assignment, the authorised administrator might be able to define additional security attributes for users.
21. Management actions due to FIA_SOS.1 (See 5.1.5):
    - The management of the metric used to verify the secrets.
22. Management actions due to FIA_SOS.2 (See 5.1.5):
    - The management of the metric used to generate the secrets.
23. Management actions due to FIA_UAU.2 (See 5.1.5):
    - Management of the authentication data by an administrator;

- management of the authentication data by the user associated with this data.

24. Management actions due to FIA_UAU.6 (See 5.1.5):
    - If an authorised administrator could request re-authentication, the management includes a re-authentication request.
25. Management actions due to FIA_UID.2 (See 5.1.5):
    - The management of the user identities.
26. Management actions due to FIA_USB.1 (See 5.1.5):
    - An authorised administrator can define default subject security attributes.
27. Management actions due to FMT_MOF.1 (See 5.1.6):
    - Managing the group of roles that can interact with the functions in the TSF;
28. Management actions due to FMT_MSA.1 (See 5.1.6):
    - Managing the group of roles that can interact with the security attributes.
29. Management actions due to FMT_MSA.3 (See 5.1.6):
    - Managing the group of roles that can specify initial values;
    - managing the permissive or restrictive setting of default values for a given access control SFP.
30. Management actions due to FMT_MTD.1 (See 5.1.6):
    - Managing the group of roles that can interact with the TSF data.
31. Management actions due to FMT_MTD.2 (See 5.1.6):
    - Managing the group of roles that can interact with the limits on the TSF data.
32. Management actions due to FMT_REV.1 (See 5.1.6):
    - Managing the group of roles that can invoke revocation of security attributes;
    - managing the lists of users, subjects, objects and other resources for which revocation is possible;
    - managing the revocation rules.
33. Management actions due to FMT_SAE.1 (See 5.1.6):
    - Managing the list of security attributes for which expiration is to be supported;
    - the actions to be taken if the expiration time has passed.
34. Management actions due to FMT_SMR.2 (See 5.1.6):
    - Managing the group of users that are part of a role;
    - managing the conditions that the roles must satisfy.
35. Management actions due to FPT_AMT.1 (See 5.1.7):
    - Management of the conditions under which abstract machine test occurs, such as during initial start-up, regular interval, or under specified conditions;
    - management of the time interval if appropriate.
36. Management actions due to FPT_ITT.1 (See 5.1.7):
    - Management of the types of modification against which the TSF should protect;
    - management of the mechanism used to provide the protection of the data in transit between different parts of the TSF.
37. Management actions due to FPT_RCV.1 (See 5.1.7):
    - Management of who can access the restore capability within the maintenance mode.

38. Management actions due to FPT_STM.1 (See 5.1.7):
    - Management of the time.
39. Management actions due to FPT_TST.1 (See 5.1.7):
    - Management of the conditions under which TSF self testing occurs, such as during initial start-up, regular interval, or under specified conditions;
    - management of the time interval if appropriate.
40. Management actions due to FRU_PRS.2 (See 5.1.8):
    - Assignment of priorities to each subject in the TSF.
41. Management actions due to FRU_RSA.2 (See 5.1.8):
    - Specifying minimum and maximum limits for a resource for groups and/or individual users and/or subjects by an administrator.
42. Management actions due to FTA_MCS.2 (See 5.1.9):
    - Management of the rules that govern the maximum allowed number of concurrent user sessions by an administrator.
43. Management actions due to FTA_SSL.1 (See 5.1.9):
    - Specification of the time of user inactivity after which lock-out occurs for an individual user;
    - specification of the default time of user inactivity after which lock-out occurs;
    - management of the events that should occur prior to unlocking the session.
44. Management actions due to FTA_SSL.2 (See 5.1.9):
    - Management of the events that should occur prior to unlocking the session.
45. Management actions due to FTA_SSL.3 (See 5.1.9):
    - Specification of the time of user inactivity after which termination of the interactive session occurs for an individual user;
    - specification of the default time of user inactivity after which termination of the interactive session occurs.
46. Management actions due to FTA_TAB.1 (See 5.1.9):
    - Maintenance of the banner by the authorised administrator.
47. Management actions due to FTA_TSE.1 (See 5.1.9):
    - Management of the session establishment conditions by the authorised administrator.
48. Management actions due to FTP_ITC.1 (See 5.1.10):
    - Configuring the actions that require trusted channel, if supported.
49. Management actions due to FTP_TRP.1 (See 5.1.10):
    - Configuring the actions that require trusted path, if supported.

to [ST ASSIGNMENT: **(the authorized identified roles)**].

FMT_MSA - Management of security attributes

This family allows authorised users control over the management of security attributes. This management might include capabilities for viewing and modifying of security attributes.

FMT_MSA.1 Management of security attributes

Management of security attributes allows authorised users (roles) to manage the specified security attributes.

FMT_MSA.1.1 The TSF shall enforce the [ST ASSIGNMENT: **(access control SFP, information flow control SFP)**] to restrict the ability to [ST SELECTION: **(change_default, query, modify, delete, [ST ASSIGNMENT: (other operations)])**] the security attributes [ST ASSIGNMENT: **(list of security attributes)**] to [ST ASSIGNMENT: **(the authorised identified roles)**].

FMT_MSA.2 Secure security attributes

Secure security attributes ensures that values assigned to security attributes are valid with respect to the secure state.

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

FMT_MSA.3 Static attribute initialisation

Static attribute initialisation ensures that the default values of security attributes are appropriately either permissive or restrictive in nature.

FMT_MSA.3.1 The TSF shall enforce the [ST ASSIGNMENT: **(access control SFP, information flow control SFP)**] to provide [ST SELECTION: **(restrictive, permissive, other property)**] default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2 The TSF shall allow the [ST ASSIGNMENT: **(the authorised identified roles)**] to specify alternative initial values to override the default values when an object or information is created.

FMT_MTD - Management of TSF data

This family allows authorised users (roles) control over the management of TSF data. Examples of TSF data include audit information, clock, system configuration and other TSF configuration parameters.

FMT_MTD.1 Management of TSF data

Management of TSF data allows authorised users to manage TSF data.

FMT_MTD.1.1 The TSF shall restrict the ability to [ST SELECTION: **(change_default, query, modify, delete, clear, [ST ASSIGNMENT: (other operations)])**] the [ST ASSIGNMENT: **(list of TSF data)**] to [ST ASSIGNMENT: **(the authorised identified roles)**].

FMT_MTD.2 Management of limits on TSF data

Management of limits on TSF data specifies the action to be taken if limits on TSF data are reached or exceeded.

FMT_MTD.2.1 The TSF shall restrict the specification of the limits for [ST ASSIGNMENT: **(list of TSF data)**] to [ST ASSIGNMENT: **(the authorised identified roles)**].

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

FMT_MTD.3 Secure TSF data

Secure TSF data ensures that values assigned to TSF data are valid with respect to the secure state.

FMT_MTD.3.1 The TSF shall ensure that only secure values are accepted for TSF data.

FMT_REV - Revocation

This family addresses revocation of security attributes for a variety of entities within a TOE.

FMT_REV.1 Revocation

Revocation provides for revocation of security attributes to be enforced at some point in time.

FMT_REV.1.1 The TSF shall restrict the ability to revoke security attributes associated with the [ST SELECTION: **(users, subjects, objects, other additional resources)**] within the TSC to [ST ASSIGNMENT: **(the authorised identified roles)**].

FMT_REV.1.2 The TSF shall enforce the rules [ST ASSIGNMENT: **(specification of revocation rules)**].

FMT_SAE - Security attribute expiration

This family addresses the capability to enforce time limits for the validity of security attributes.

FMT_SAE.1 Time-limited authorisation

Time-limited authorisation provides the capability for an authorised user to specify an expiration time on specified security attributes.

FMT_SAE.1.1 The TSF shall restrict the capability to specify an expiration time for [ST ASSIGNMENT: **(list of security attributes for which expiration is to be supported)**] to [ST ASSIGNMENT: **(the authorised identified roles)**].

FMT_SAE.1.2 For each of these security attributes, the TSF shall be able to [ST ASSIGNMENT: **(list of actions to be taken for each security attribute)**] after the expiration time for the indicated security attribute has passed.

FMT_SMR - Security management roles

This family is intended to control the assignment of different roles to users. The capabilities of these roles with respect to security management are described in the other families in this class.

FMT_SMR.2 Restrictions on security roles

Restrictions on security roles specifies that in addition to the specification of the roles, there are rules that control the relationship between the roles.

FMT_SMR.2.1 The TSF shall maintain the following roles:

Administrator In charge of installing, configuring, management of the TSF. This role is assumed to perform cryptographic initialization or management functions (e.g., module initialization, input/output of cryptographic keys and CSPs).

Operator In charge of performing TOE backup and recovery.

User The user operates over the provided cryptographic functionality. Semantics of these functions is irrelevant to this TOE.

Auditor Review and management of audit log.

The TSF may support other roles or sub-roles in addition to the roles specified above, under the following premises:

- All plaintext secret and private keys and unprotected CSPs shall be zeroized when entering or exiting the role.
- The separation of duties defined above is not violated under the new role capabilities.

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 **static separation of duties for defined roles as defined by [Nat00]** are satisfied.

5.1.7 FPT - Protection of the TSF

This class contains families of functional requirements that relate to the integrity and management of the mechanisms that provide the TSF (independent of TSPspecifics) and to the integrity of TSF data (independent of the specific contents of the TSP data). In some sense, families in this class may appear to duplicate components in the FDP (User data protection) class; they may even be implemented using the same mechanisms. However, FDP focuses on user data protection, while FPT focuses on TSF data protection. In fact, components from the FPT class are necessary to provide requirements that the SFPs in the TOE cannot be tampered with or bypassed.

FPT_AMT - Underlying abstract machine test

This family defines requirements for the TSF to perform testing to demonstrate the security assumptions made about the underlying abstract machine upon which the TSF relies. This "abstract" machine could be a hardware/firmware platform, or it could be some known and assessed hardware/software combination acting as a virtual machine.

FPT_AMT.1 Abstract machine testing

Abstract machine testing, provides for testing of the underlying abstract machine.

FPT_AMT.1.1 The TSF shall run a suite of tests **during initial start-up, and every time a component of the abstract machine is loaded or invoked, for example when dynamically loading libraries,** to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the TSF.

FPT_FLS - Fail secure

The requirements of this family ensure that the TOE will not violate its TSP in the event of identified categories of failures in the TSF.

FPT_FLS.1 Failure with preservation of secure state

Failure with preservation of secure state, which requires that the TSF preserve a secure state in the face of the identified failures.

FPT_FLS.1.1 The TSF shall preserve a secure state when the following types of failures occur: [ST ASSIGNMENT: **(list of types of failures in the TSF)**].

FPT_ITI - Integrity of exported TSF data

This family defines the rules for the protection, from unauthorised modification, of TSF data during transmission between the TSF and a remote trusted IT product. This data could, for example, be TSF critical data such as passwords, keys, audit data, or TSF executable code.

FPT_ITI.1 Inter-TSF detection of modification

Inter-TSF detection of modification, provides the ability to detect modification of TSF data during transmission between the TSF and a remote trusted IT product, under the assumption that the remote trusted IT product is cognisant of the mechanism used.

FPT_ITI.1.1 The TSF shall provide the capability to detect modification of all TSF data during transmission between the TSF and a remote trusted IT product within the following metric: [ST ASSIGNMENT: **(a defined modification metric)**].

FPT_ITI.1.2 The TSF shall provide the capability to verify the integrity of all TSF data transmitted between the TSF and a remote trusted IT product and perform [ST ASSIGNMENT: **(action to be taken)**] if modifications are detected.

FPT_ITT - Internal TOE TSF data transfer

This family provides requirements that address protection of TSF data when it is transferred between separate parts of a TOE across an internal channel.

FPT_ITT.1 Basic internal TSF data transfer protection

Basic internal TSF data transfer protection, requires that TSF data be protected when transmitted between separate parts of the TOE.

FPT_ITT.1.1 The TSF shall protect TSF data from [ST SELECTION: **(disclosure, modification)**] when it is transmitted between separate parts of the TOE.

FPT_RCV - Trusted recovery

The requirements of this family ensure that the TSF can determine that the TOE is started up without protection compromise and can recover without protection compromise after discontinuity of operations. This family is important because the start-up state of the TSF determines the protection of subsequent states.

FPT_RCV.1 Manual recovery

Manual recovery, allows a TOE to only provide mechanisms that involve human intervention to return to a secure state.

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

FPT_RCV.4 Function recovery

Function recovery, provides for recovery at the level of particular SFs, ensuring either successful completion or rollback of TSF data to a secure state.

FPT_RCV.4.1 The TSF shall ensure that [ST ASSIGNMENT: **(list of SFs and failure scenarios)**] have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

FPT_RVM - Reference mediation

The requirements of this family address the "always invoked" aspect of a traditional reference monitor. The goal of this family is to ensure, with respect to a given SFP, that all actions requiring policy enforcement are validated by the TSF against the SFP. If the portion of the TSF that enforces the SFP also meets the requirements of appropriate components from FPT_SEP (Domain separation) and ADV_INT (TSF internals), then that portion of the TSF provides a "reference monitor" for that SFP.

FPT_RVM.1 Non-bypassability of the TSP

Non-bypassability of the TSP, which requires non-bypassability for all SFPs in 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 - Domain separation

The components of this family ensure that at least one security domain is available for the TSF's own execution and that the TSF is protected from external interference and tampering (e.g. by modification of TSF code or data structures) by untrusted subjects. Satisfying the requirements of this family makes the TSF selfprotecting, meaning that an untrusted subject cannot modify or damage the TSF.

FPT_SEP.3 Complete reference monitor

Complete reference monitor, requires that there be distinct domain(s) for TSP enforcement, a domain for the remainder of the TSF, as well as domains for the non-TSF portions of the TOE.

FPT_SEP.3.1 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.3.2 The TSF shall enforce separation between the security domains of subjects in the TSC.

FPT_SEP.3.3 The TSF shall maintain the part of the TSF that enforces the access control and/ or information flow control SFPs in a security domain for its own execution that protects them from interference and tampering by the remainder of the TSF and by subjects untrusted with respect to the TSP.

FPT_SSP - State synchrony protocol

Distributed systems may give rise to greater complexity than monolithic systems through the potential for differences in state between parts of the system, and through delays in communication. In most cases synchronisation of state between distributed functions involves an exchange protocol, not a simple action. When malice exists in the distributed environment of these protocols, more complex defensive protocols are required.

FPT_SSP.2 Mutual trusted acknowledgement

Mutual trusted acknowledgement requires mutual acknowledgment of the data exchange.

FPT_SSP.2.1 The TSF shall acknowledge, when requested by another part of the TSF, the receipt of an unmodified TSF data transmission.

FPT_SSP.2.2 The TSF shall ensure that the relevant parts of the TSF know the correct status of transmitted data among its different parts, using acknowledgements.

FPT_STM - Time stamps

This family addresses requirements for a reliable time stamp function within a TOE.

FPT_STM.1 Reliable time stamps

Reliable time stamps, which requires that the TSF provide reliable time stamps for TSF functions.

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

FPT_TDC - Inter-TSF TSF data consistency

In a distributed or composite system environment, a TOE may need to exchange TSF data (e.g. the SFP-attributes associated with data, audit information, identification information) with another trusted IT product. This family defines the requirements for sharing and consistent interpretation of these attributes between the TSF of the TOE and a different trusted IT product.

FPT_TDC.1 Inter-TSF basic TSF data consistency

Inter-TSF basic TSF data consistency requires that the TSF provide the capability to ensure consistency of attributes between TSFs.

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret [ST ASSIGNMENT: **(list of TSF data types)**] when shared between the TSF and another trusted IT product.

FPT_TDC.1.2 The TSF shall use [ST ASSIGNMENT: **(list of interpretation rules to be applied by the TSF)**] when interpreting the TSF data from another trusted IT product.

FPT_TRC - Internal TOE TSF data replication consistency

The requirements of this family are needed to ensure the consistency of TSF data when such data is replicated internal to the TOE. Such data may become inconsistent if the internal channel between parts of the TOE becomes inoperative. If the TOE is internally structured as a network and parts of the TOE network connections are broken, this may occur when parts become disabled.

FPT_TRC.1 Internal TSF consistency

Internal TSF consistency, which requires that the TSF ensure the consistency of TSF data that is replicated in multiple locations.

FPT_TRC.1.1 The TSF shall ensure that TSF data is consistent when replicated between parts of the TOE.

FPT_TRC.1.2 When parts of the TOE containing replicated TSF data are disconnected, the TSF shall ensure the consistency of the replicated TSF data upon reconnection before processing any requests for [ST ASSIGNMENT: **(list of SFs dependent on TSF data replication consistency)**].

FPT_TST - TSF self test

The family defines the requirements for the self-testing of the TSF with respect to some expected correct operation. Examples are interfaces to enforcement functions, and sample arithmetical operations on critical parts of the TOE. These tests can be carried out at start-up, periodically, at the request of the authorised user, or when other conditions are met. The actions to be taken by the TOE as the result of self testing are defined in other families.
The requirements of this family are also needed to detect the corruption of TSF executable code (i.e. TSF software) and TSF data by various failures that do not necessarily stop the TOE's operation (which would be handled by other families). These checks must be performed because these failures may not necessarily be prevented. Such failures can occur either because of unforeseen failure modes or associated oversights in the design of hardware, firmware, or software, or because of malicious corruption of the TSF due to inadequate logical and/or physical protection.

FPT_TST.1 TSF testing

TSF testing, provides the ability to test the TSF's correct operation. These tests may be performed at start-up, periodically, at the request of the authorised user, or when other conditions are met. It also provides the ability to verify the integrity of TSF data and executable code.

FPT_TST.1.1 A TSF shall perform power-up self-tests and conditional self-tests to ensure that the module is functioning properly. Power-up self-tests shall be performed when the TSF is powered up. Conditional self-tests shall be performed when an applicable security function or operation is invoked (i.e., security functions for which self-tests are required).

The TSF shall run a suite of additional self tests [ST SELECTION: **(during initial start-up, periodically during normal operation, at the request of the authorized user, at the conditions [ST ASSIGNMENT: (conditions under which self test should occur)])**] to demonstrate the correct operation of the TSF.

If a TSF fails a self-test, the module shall enter an error state and output an error indicator via the status output interface. The TSF shall not perform any cryptographic operations while in an error state. All data output via the data output interface shall be inhibited when an error state exists.

Power-Up Tests

Power-up tests shall be performed by a TSF when the TOE is powered up (after being powered off, reset, rebooted, etc.). The power-up tests shall be initiated automatically and shall not require operator intervention. When the power-up tests are completed, the results (i.e., indications of success or failure) shall be output via the status output interface. All data output via the data output interface shall be inhibited when the power-up tests are performed.

In addition to performing the power-up tests when powered up, a TSF shall permit operators to initiate the tests on demand for periodic testing of the TSF. Resetting, rebooting, and power cycling are acceptable means for the on-demand initiation of power-up tests.

A TSF shall perform the following power-up tests:

- Cryptographic algorithm test.
- Software/firmware integrity test.
- Critical functions test.
- Statistical random number generator tests.

Cryptographic algorithm test. A cryptographic algorithm test using a known answer shall be conducted for all modes (e.g., encryption, decryption, authentication, and deterministic random number generation) of each Approved cryptographic algorithm implemented by a TSF. A known-answer test involves operating the cryptographic algorithm on data for which the correct output is already known and comparing the calculated output with the previously generated output (the known answer). If the calculated output does not equal the known answer, the known-answer test shall fail.

Cryptographic algorithms whose outputs vary for a given set of inputs shall be tested using a known-answer test or shall be tested using a pair-wise consistency test (specified below). Message digest algorithms shall have an independent known-answer test or the known-answer test shall be included with the associated cryptographic algorithm test.

If a TSF includes two independent implementations of the same cryptographic algorithm, then:

- the known-answer test may be omitted,
- the outputs of two implementations shall be continuously compared, and
- if the outputs of two implementations are not equal, the cryptographic algorithm test shall fail.

Software/firmware integrity test. A software/firmware integrity test using an error detection code or Approved authentication technique (e.g., an Approved message authentication code or digital signature algorithm) shall be applied to all validated software and firmware components within a TSF when the TOE is powered up. The software/firmware integrity test is not required for any software and firmware components excluded from the security requirements of this PP. If the calculated result does not equal the previously generated result, the software/firmware test shall fail.

If an error detection code is used, it shall be at least 16 bits in length.

Critical functions test. Other security functions critical to the secure operation of a TSF shall be tested when the TOE is powered up as part of the power-up tests. Other critical security functions performed under specific conditions shall be tested as conditional tests.

Statistical random number generator tests. If statistical random number generator tests are required (i.e., depending on the security level), a TSF employing RNGs

Length of Run	Required Interval
1	2,315 - 2,685
2	1,114 - 1,386
3	527 - 723
4	240 - 384
5	103 - 209
6+	103 - 209

Table 5.1: Required intervals for length of runs test

shall perform the following statistical tests for randomness. A single bit stream of 20,000 consecutive bits of output from each RNG shall be subjected to the following four tests: monobit test, poker test, runs test, and long runs test.

The monobit test

1. Count the number of ones in the $20,000$ bit stream. Denote this quantity by X.
2. The test is passed if $9,725 < X < 10,275$.

The poker test

1. Divide the $20,000$ bit stream into $5,000$ consecutive 4 bit segments. Count and store the number of occurrences of the 16 possible 4 bit values. Denote $f(i)$ as the number of each 4 bit value i, where $0 \leq i \leq 15$.
2. Evaluate the following:
$$X = \frac{16}{5000} \times \left(\sum_{i=0}^{15} [f(i)]^2\right) - 5000$$
3. The test is passed if $2.16 < X < 46.17$.

The runs test

1. A run is defined as a maximal sequence of consecutive bits of either all ones or all zeros that is part of the $20,000$ bit sample stream. The incidences of runs (for both consecutive zeros and consecutive ones) of all lengths (≥ 1) in the sample stream should be counted and stored.
2. The test is passed if the runs that occur (of lengths 1 through 6) are each within the corresponding interval specified in table 5.1. This must hold for both the zeros and ones (i.e., all 12 counts must lie in the specified interval). For the purposes of this test, runs of greater than 6 are considered to be of length 6.

The long runs test

1. A long run is defined to be a run of length 26 or more (of either zeros or ones).
2. On the sample of $20,000$ bits, the test is passed if there are no long runs.

Conditional Tests

Conditional tests tests shall be performed by a TSF when the conditions specified for the following tests occur: pair-wise consistency test, software/firmware load test, manual key entry test, continuous random number generator test, and bypass test.

Pair-wise consistency test (for public and private keys). If a TSF generates public or private keys, then the following pair-wise consistency tests for public and private keys shall be performed:

1. If the keys are used to perform key transport or encryption, then the public key shall encrypt a plaintext value. The resulting ciphertext value shall be compared to the original plaintext value. If the two values are equal, then the test shall fail. If the two values differ, then the private key shall be used to decrypt the ciphertext and the resulting value shall be compared to the original plaintext value. If the two values are not equal, the test shall fail.
2. If the keys are used to perform key agreement, then the TSF shall create a second, compatible key pair. The TSF shall perform both sides of the key agreement algorithm and shall compare the resulting shared values. If the shared values are not equal, the test shall fail.
3. If the keys are used to perform the calculation and verification of digital signatures, then the consistency of the keys shall be tested by the calculation and verification of a digital signature. If the digital signature cannot be verified, the test shall fail.

Software/firmware load test. If software or firmware components can be externally loaded into a TSF, then the following software/firmware load tests shall be performed:

1. An Approved authentication technique (e.g., an Approved message authentication code, digital signature algorithm, or HMAC) shall be applied to all validated software and firmware components when the components are externally loaded into a TSF. The software/firmware load test is not required for any software and firmware components excluded from the security requirements of this PP.
2. The calculated result shall be compared with a previously generated result. If the calculated result does not equal the previously generated result, the software/firmware load test shall fail.

Manual key entry test. If cryptographic keys or key components are manually entered into a TSF, then the following manual key entry tests shall be performed:

1. The cryptographic key or key components shall have an EDC applied, or shall be entered using duplicate entries.
2. If an EDC is used, the EDC shall be at least 16 bits in length.
3. If the EDC cannot be verified, or the duplicate entries do not match, the test shall fail.

Continuous random number generator test. If a TSF employs Approved or non-Approved RNGs in an Approved mode of operation, the TSF shall perform the following continuous random number generator test on each RNG that tests for failure to a constant value.

1. If each call to a RNG produces blocks of n bits (where $n > 15$), the first n-bit block generated after power-up, initialization, or reset shall not be used, but shall be saved for comparison with the next n-bit block to be generated. Each subsequent generation of an n-bit block shall be compared with the previously generated block. The test shall fail if any two compared n-bit blocks are equal.
2. If each call to a RNG produces fewer than 16 bits, the first n bits generated after power-up, initialization, or reset (for some $n > 15$) shall not be used, but shall be saved for comparison with the next n generated bits. Each subsequent generation of n bits shall be compared with the previously generated n bits. The test fails if any two compared n-bit sequences are equal.

Bypass test. If a TSF implements a bypass capability where the services may be provided without cryptographic processing (e.g., transferring plaintext through the TSF), then the following bypass tests shall be performed to ensure that a single point of failure of TOE components will not result in the unintentional output of plaintext:

1. A TSF shall test for the correct operation of the services providing cryptographic processing when a switch takes place between an exclusive bypass service and an exclusive cryptographic service.
2. If a TSF can automatically alternate between a bypass service and a cryptographic service, providing some services with cryptographic processing and some services without cryptographic processing, then the TSF shall test for the correct operation of the services providing cryptographic processing when the mechanism governing the switching procedure is modified (e.g., an IP address source/destination table).

FPT_TST.1.2 The TSF shall provide authorised users with the capability to verify the integrity of TSF data.

FPT_TST.1.3 The TSF shall provide authorised users with the capability to verify the integrity of stored TSF executable code.

5.1.8 FRU - Resource utilisation

This class provides three families that support the availability of required resources such as processing capability and/or storage capacity. The family Fault Tolerance provides protection against unavailability of capabilities caused by failure of the TOE. The family Priority of Service ensures that the resources will be allocated to the more important or time-critical tasks and cannot be monopolised by lower priority tasks. The family Resource Allocation provides limits on the use of available resources, therefore preventing users from monopolising the resources.

FRU_PRS - Priority of service

The requirements of this family allow the TSF to control the use of resources within the TSC by users and subjects such that high priority activities within the TSC will always be accomplished without undue interference or delay caused by low priority activities.

FRU_PRS.2 Full priority of service

Full priority of service provides priorities for a subject's use of all of the resources within the TSC.

FRU_PRS.2.1 The TSF shall assign a priority to each subject in the TSF.

FRU_PRS.2.2 The TSF shall ensure that each access to all shareable resources shall be mediated on the basis of the subjects assigned priority.

FRU_RSA - Resource allocation

The requirements of this family allow the TSF to control the use of resources by users and subjects such that denial of service will not occur because of unauthorised monopolisation of resources.

FRU_RSA.2 Minimum and maximum quotas

Minimum and maximum quotas provides requirements for quota mechanisms that ensure that users and subjects will always have at least a minimum of a specified resource and that they will not be able to monopolise a controlled resource.

FRU_RSA.2.1 The TSF shall enforce maximum quotas of the following resources [ST ASSIGNMENT: **(controlled resources)**] that [ST SELECTION: **(individual user, defined group of users)**] can use [ST SELECTION: **(simultaneously, over a specified period of time)**].

FRU_RSA.2.2 The TSF shall ensure the provision of minimum quantity of each [ST ASSIGNMENT: **(controlled resource)**] that is available for [ST SELECTION: **(an individual user, defined group of users, subjects)**] to use [ST SELECTION: **(simultaneously, over a specified period of time)**]

5.1.9 FTA - TOE access

This family specifies functional requirements for controlling the establishment of a user's session.

FTA_MCS - Limitation on multiple concurrent sessions

This family defines requirements to place limits on the number of concurrent sessions that belong to the same user.

FTA_MCS.2 Per user attribute limitation on multiple concurrent sessions

Per user attribute limitation on multiple concurrent sessions extends FTA_MCS.1 by requiring the ability to specify limitations on the number of concurrent sessions based on the related security attributes.

FTA_MCS.2.1 The TSF shall restrict the maximum number of concurrent sessions that belong to the same user according to the rules [ST ASSIGNMENT: **(rules for the number of maximum concurrent sessions)**].

FTA_MCS.2.2 The TSF shall enforce, by default, a limit of [ST ASSIGNMENT: **(default number)**] sessions per user.

FTA_SSL - Session locking

This family defines requirements for the TSF to provide the capability for TSFinitiated and user-initiated locking and unlocking of interactive sessions.

FTA_SSL.1 TSF-initiated session locking

TSF-initiated session locking includes system initiated locking of an interactive session after a specified period of user inactivity.

FTA_SSL.1.1 The TSF shall lock an interactive session after [ST ASSIGNMENT: **(time interval of user inactivity)**] by:

- clearing or overwriting display devices, making the current contents unreadable;
- disabling any activity of the user's data access/display devices other than unlocking the session.

FTA_SSL.1.2 The TSF shall require the following events to occur prior to unlocking the session: [ST ASSIGNMENT: **(events to occur)**].

FTA_SSL.2 User-initiated locking

User-initiated locking provides capabilities for the user to lock and unlock the user's own interactive sessions.

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

- clearing or overwriting display devices, making the current contents unreadable;
- disabling any activity of the user's data access/display devices other than unlocking the session.

FTA_SSL.2.2 The TSF shall require the following events to occur prior to unlocking the session: [ST ASSIGNMENT: **(events to occur)**].

FTA_SSL.3 TSF-initiated termination

TSF-initiated termination provides requirements for the TSF to terminate the session after a period of user inactivity.

FTA_SSL.3.1 The TSF shall terminate an interactive session after a [ST ASSIGNMENT: **(time interval of user inactivity)**].

FTA_TAB - TOE access banners

This family defines requirements to display a configurable advisory warning message to users regarding the appropriate use of the TOE.

FTA_TAB.1 Default TOE access banners

Default TOE access banners provides the requirement for a TOE Access Banner. This banner is displayed prior to the establishment dialogue for a session.

FTA_TAB.1.1 Before establishing a user session, the TSF shall display an advisory warning message regarding unauthorised use of the TOE.

FTA_TAH - TOE access history

This family defines requirements for the TSF to display to a user, upon successful session establishment, a history of successful and unsuccessful attempts to access the user's account.

FTA_TAH.1 TOE access history

TOE access history provides the requirement for a TOE to display information related to previous attempts to establish a session.

FTA_TAH.1.1 Upon successful session establishment, the TSF shall display the [ST SELECTION: **(date, time, method, location)**] of the last successful session establishment to the user.

FTA_TAH.1.2 Upon successful session establishment, the TSF shall display the [ST SELECTION: **(date, time, method, location)**] of the last unsuccessful attempt to session establishment and the number of unsuccessful attempts since the last successful session establishment.

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

FTA_TSE - TOE session establishment

This family defines requirements to deny a user permission to establish a session with the TOE.

FTA_TSE.1 TOE session establishment

TOE session establishment provides requirements for denying users access to the TOE based on attributes.

FTA_TSE.1.1 The TSF shall be able to deny session establishment based on [ST ASSIGNMENT: **(attributes)**].

5.1.10 FTP - Trusted path/channels

Families in this class provide requirements for a trusted communication path between users and the TSF, and for a trusted communication channel between the TSF and other trusted IT products. Trusted paths and channels have the following general characteristics:

- The communications path is constructed using internal and external communications channels (as appropriate for the component) that isolate an identified subset of TSF data and commands from the remainder of the TSF and user data.
- Use of the communications path may be initiated by the user and/or the TSF (as appropriate for the component)
- The communications path is capable of providing assurance that the user is communicating with the correct TSF, and that the TSF is communicating with the correct user (as appropriate for the component)

FTP_ITC - Inter-TSF trusted channel

This family defines requirements for the creation of a trusted channel between the TSF and other trusted IT products for the performance of security critical operations. This family should be included whenever there are requirements for the secure communication of user or TSF data between the TOE and other trusted IT products.

FTP_ITC.1 Inter-TSF trusted channel

Inter-TSF trusted channel requires that the TSF provide a trusted communication channel between itself and another trusted IT product.

FTP_ITC.1.1 The TSF shall provide a communication channel between itself and a remote trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

FTP_ITC.1.2 The TSF shall permit [ST SELECTION: **(the TSF, the remote trusted IT product)**] to initiate communication via the trusted channel.

FTP_ITC.1.3 The TSF shall initiate communication via the trusted channel for [ST ASSIGNMENT: **(list of functions for which a trusted channel is required)**].

FTP_TRP - Trusted path

This family defines the requirements to establish and maintain trusted communication to or from users and the TSF. A trusted path may be required for any security-relevant interaction. Trusted path exchanges may be initiated by a user during an interaction with the TSF, or the TSF may establish communication with the user via a trusted path.

FTP_TRP.1 Trusted path

Trusted path requires that a trusted path between the TSF and a user be provided for a set of events defined by a PP/ST author. The user and/or the TSF may have the ability to initiate the trusted path.

FTP_TRP.1.1 The TSF shall provide a communication path between itself and [ST SELECTION: **(remote, local)**] users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from modification or disclosure.

FTP_TRP.1.2 The TSF shall permit [ST SELECTION: **(the TSF, local users, remote users)**] to initiate communication via the trusted path.

FTP_TRP.1.3 The TSF shall require the use of the trusted path for [ST SELECTION: **(initial user authentication, [ST ASSIGNMENT: (other services for which trusted path is required)])**].

5.2 TOE Security Assurance Requirements

Actual assurance requirements are dictated by an Organizational General Security Policy, (See 3.7.7) that takes into consideration areas of concern common to any PKI element, while being consistent with the security policy models provided and requested by this PP.

This Assurance Requirements set meets [the99d] EAL4 augmented. The actual augmentations are defined at table 5.2.

Component	Evaluation Assurance Level	Augmentation Degree
ACM_AUT.2 (See 5.2.1)	EAL4 augmented	EAL6
ACM_CAP.4 (See 5.2.1)	EAL4	EAL4
ACM_SCP.3 (See 5.2.1)	EAL4 augmented	EAL5
ADO_DEL.2 (See 5.2.2)	EAL4	EAL4
ADO_IGS.1 (See 5.2.2)	EAL4	EAL4
ADV_FSP.3 (See 5.2.3)	EAL4 augmented	EAL5
ADV_HLD.3 (See 5.2.3)	EAL4 augmented	EAL5
ADV_IMP.3 (See 5.2.3)	EAL4 augmented	EAL6
ADV_INT.3 (See 5.2.3)	EAL4 augmented	EAL7
ADV_LLD.1 (See 5.2.3)	EAL4	EAL4
ADV_RCR.2 (See 5.2.3)	EAL4 augmented	EAL5
ADV_SPM.2 (See 5.2.3)	EAL4 augmented	EAL4 augmented
AGD_ADM.1 (See 5.2.4)	EAL4	EAL4
AGD_USR.1 (See 5.2.4)	EAL4	EAL4
ALC_DVS.2 (See 5.2.5)	EAL4 augmented	EAL6
ALC_FLR.3 (See 5.2.5)	EAL4 augmented	EAL4 augmented
ALC_LCD.3 (See 5.2.5)	EAL4 augmented	EAL7
ALC_TAT.2 (See 5.2.5)	EAL4 augmented	EAL5
ATE_COV.2 (See 5.2.6)	EAL4	EAL4
ATE_DPT.3 (See 5.2.6)	EAL4 augmented	EAL7
ATE_FUN.2 (See 5.2.6)	EAL4 augmented	EAL6
ATE_IND.2 (See 5.2.6)	EAL4	EAL4
AVA_CCA.1 (See 5.2.7)	EAL4 augmented	EAL5
AVA_MSU.2 (See 5.2.7)	EAL4	EAL4
AVA_SOF.1 (See 5.2.7)	EAL4	EAL4
AVA_VLA.2 (See 5.2.7)	EAL4	EAL4

Table 5.2: EAL level

5.2.1 ACM - Configuration management

Configuration management (CM) is one method or means for establishing that the functional requirements and specifications are realised in the implementation of the TOE. CM meets these objectives by requiring discipline and control in the processes of refinement and modification of the TOE and the related information. CM systems are put in place to ensure the integrity of the portions of the TOE that they control, by providing a method of tracking any changes, and by ensuring that all changes are authorised.

ACM_AUT - CM automation

The objective of introducing automated CM tools is to increase the effectiveness of the CM system. While both automated and manual CM systems can be bypassed, ignored, or prove insufficient to prevent unauthorised modification, automated systems are less susceptible to human error or negligence.

ACM_AUT.2 Complete CM automation

In development environments where the configuration items are complex or are being developed by multiple developers, it is difficult to control changes without the support of automated tools. In particular, these automated tools need to be able to support the numerous changes that occur during development and ensure that those changes are authorised. It is the objective of this component to ensure that all configuration items are controlled through automated means.

Providing an automated means of ascertaining changes between versions of the TOE and identifying which configuration items are affected by modifications to other configuration items assists in determining the impact of the changes between successive versions of the TOE. This in turn can provide valuable information in determining whether changes to the TOE result in all configuration items being consistent with one another.

ACM_AUT.2.1C The CM system shall provide an automated means by which only authorised changes are made to the TOE implementation representation, and to all other configuration items.

ACM_AUT.2.1D The developer shall use a CM system.

ACM_AUT.2.2C The CM system shall provide an automated means to support the generation of the TOE.

ACM_AUT.2.2D The developer shall provide a CM plan.

ACM_AUT.2.3C The CM plan shall describe the automated tools used in the CM system.

ACM_AUT.2.4C The CM plan shall describe how the automated tools are used in the CM system.

ACM_AUT.2.5C The CM system shall provide an automated means to ascertain the changes between the TOE and its preceding version.

ACM_AUT.2.6C The CM system shall provide an automated means to identify all other configuration items that are affected by the modification of a given configuration item.

ACM_CAP - CM capabilities

The capabilities of the CM system address the likelihood that accidental or unauthorised modifications of the configuration items will occur. The CM system should ensure the integrity of the TOE from the early design stages through all subsequent maintenance efforts.

ACM_CAP.4 Generation support and acceptance procedures

A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

Providing controls to ensure that unauthorised modifications are not made to the TOE, and ensuring proper functionality and use of the CM system, helps to maintain the integrity of the TOE.

The purpose of acceptance procedures is to confirm that any creation or modification of configuration items is authorised.

ACM_CAP.4.10C The CM system shall provide measures such that only authorised changes are made to the configuration items.

ACM_CAP.4.11C The CM system shall support the generation of the TOE.

ACM_CAP.4.12C The acceptance plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.

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

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

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

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

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

ACM_CAP.4.3D The developer shall provide CM documentation.

ACM_CAP.4.4C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.4.5C The CM documentation shall describe the method used to uniquely identify the configuration items.

ACM_CAP.4.6C The CM system shall uniquely identify all configuration items.

ACM_CAP.4.7C The CM plan shall describe how the CM system is used.

ACM_CAP.4.8C The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.4.9C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_SCP - CM scope

The objective of this family is to ensure that all necessary TOE configuration items are tracked by the CM system. This helps to ensure that the integrity of these configuration items is protected through the capabilities of the CM system.

ACM_SCP.3 Development tools CM coverage

A CM system can control changes only to those items that have been placed under CM. Placing the TOE implementation representation, design, tests, user and administrator documentation, and CM documentation under CM provides assurance that they have been modified in a controlled manner with proper authorisations.
The ability to track security flaws under CM ensures that security flaw reports are not lost or forgotten, and allows a developer to track security flaws to their resolution.
Development tools play an important role in ensuring the production of a quality version of the TOE. Therefore, it is important to control modifications to these tools.

ACM_SCP.3.1C The CM documentation shall show that the CM system, as a minimum, tracks the following: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, CM documentation, security flaws, and development tools and related information.

ACM_SCP.3.1D The developer shall provide CM documentation.

ACM_SCP.3.2C The CM documentation shall describe how configuration items are tracked by the CM system.

5.2.2 ADO - Delivery and operation

Delivery and operation provides requirements for correct delivery, installation, generation, and start-up of the TOE.

ADO_DEL - Delivery

The requirements for delivery call for system control and distribution facilities and procedures that provide assurance that the recipient receives the TOE that the sender intended to send, without any modifications. For a valid delivery, what is received must correspond precisely to the TOE master copy, thus avoiding any tampering with the actual version, or substitution of a false version.

ADO_DEL.2 Detection of modification

> **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 userŠs site.
>
> **ADO_DEL.2.1D** The developer shall document procedures for delivery of the TOE or parts of it to the user.
>
> **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 developerŠs master copy and the version received at the user site.
>
> **ADO_DEL.2.2D** The developer shall use the delivery procedures.
>
> **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 userŠs site.

ADO_IGS - Installation, generation and start-up

Installation, generation, and start-up procedures are useful for ensuring that the TOE has been installed, generated, and started up in a secure manner as intended by the developer. The requirements for installation, generation and start-up call for a secure transition from the TOE's implementation representation being under configuration control to its initial operation in the user environment.

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

> **ADO_IGS.1.1C** The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.
>
> **ADO_IGS.1.1D** The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

5.2.3 ADV - Development

The development class encompasses four families of requirements for representing the TSF at various levels of abstraction from the functional interface to the implementation representation. The development class also includes a family of requirements for a correspondence mapping between the various TSF representations, ultimately requiring a demonstration of correspondence from the least abstract representation through all intervening representations to the TOE summary specification provided in the ST. In addition, there is a family of requirements for a TSP model, and for correspondence mappings between the TSP, the TSP model, and the functional specification. Finally, there is a family of requirements on the internal structure of the TSF, which covers aspects such as modularity, layering, and minimisation of complexity.

ADV_FSP - Functional specification

The functional specification is a high-level description of the user-visible interface and behaviour of the TSF. It is an instantiation of the TOE security functional requirements. The functional specification has to show that all the TOE security functional requirements are addressed.

ADV_FSP.3 Semiformal functional specification

ADV_FSP.3.1C The functional specification shall describe the TSF and its external interfaces using a semiformal style, supported by informal, explanatory text where appropriate.

ADV_FSP.3.1D The developer shall provide a functional specification.

ADV_FSP.3.2C The functional specification shall be internally consistent.

ADV_FSP.3.3C The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing complete details of all effects, exceptions and error messages.

ADV_FSP.3.4C The functional specification shall completely represent the TSF.

ADV_FSP.3.5C The functional specification shall include rationale that the TSF is completely represented.

ADV_FSP.3.6C Finite State Model

The operation of a TSF shall be specified using a finite state model (or equivalent) represented by a state transition diagram and/or a state transition table.

The state transition diagram and/or state transition table includes:

- all operational and error states of a TSF,
- the corresponding transitions from one state to another,
- the input events that cause transitions from one state to another, and
- the output events resulting from transitions from one state to another.

A TSF shall include the following operational and error states:

Power on/off states. States for primary, secondary, or backup power. These states may distinguish between power sources being applied to a TSF.

Crypto officer states. States in which the crypto officer services are performed (e.g., cryptographic initialization and key management).

Key/CSP entry states. States for entering cryptographic keys and CSPs into the TSF.

User states. States in which authorized users obtain security services, perform cryptographic operations, or perform other Approved or non-Approved functions.

Self-test states. States in which the TSF is performing self-tests.

Error states. States when the TSF has encountered an error (e.g., failed a self-test or attempted to encrypt when missing operational keys or CSPs). Error states may include "hard" errors that indicate an equipment malfunction and that may require maintenance, service or repair of the TSF, or recoverable "soft" errors that may require initialization or resetting of the module. Recovery from error states shall be possible except for those caused by hard errors that require maintenance, service, or repair of the TSF.

A TSF may contain other states including, but not limited to, the following:

Bypass states. States in which a bypass capability is activated and services are provided without cryptographic processing (e.g., transferring plaintext through the TSF).

Maintenance states. States for maintaining and servicing a TSF, including physical and logical maintenance testing. If a TSF contains a maintenance role, then a maintenance state shall be included.

Documentation shall include a representation of the finite state model (or equivalent) using a state transition diagram and/or state transition table that shall specify:

- all operational and error states of a TSF,
- the corresponding transitions from one state to another,

ADV_FSP.3.7C TSF Ports and Interfaces

A TSF shall restrict all information flow and physical access points to physical ports and logical interfaces that define all entry and exit points to and from the module. The TSF interfaces shall be logically distinct from each other although they may share one physical port (e.g., input data may enter and output data may exit via the same port) or may be distributed over one or more physical ports (e.g., input data may enter via both a serial and a parallel port). An Application Program Interface (API) of a software component of a TSF may be defined as one or more logical interfaces(s).

A TSF shall have the following four logical interfaces ("input" and "output" are indicated from the perspective of the module):

Data input interface. All data (except control data entered via the control input interface) that is input to and processed by a TSF (including plaintext data, ciphertext data, cryptographic keys and CSPs, authentication data, and status information from another module) shall enter via the "data input" interface.

Data output interface. All data (except status data output via the status output interface) that is output from a TSF (including plaintext data, ciphertext data, cryptographic keys and CSPs, authentication data, and control information for another module) shall exit via the "data output" interface. All data output via the data output interface shall be inhibited when an error state exists and during self-tests (see 5.1.7).

Control input interface. All input commands, signals, and control data (including function calls and manual controls such as switches, buttons, and keyboards) used to control the operation of a TSF shall enter via the "control input" interface.

Status output interface. All output signals, indicators, and status data (including return codes and physical indicators such as Light Emitting Diodes and displays) used to indicate the status of a TSF shall exit via the "status output" interface.

All external electrical power that is input to a TSF (including power from an external power source or batteries) shall enter via a power port. A power port is not required when all power is provided or maintained internally to the cryptographic boundary of the TSF (e.g., an internal battery).

The TSF shall distinguish between data and control for input and data and status for output. All input data entering the TSF via the "data input" interface shall only pass through the input data path. All output data exiting the TSF via the "data output" interface shall only pass through the output data path. The output data path shall be logically disconnected from the circuitry and processes while performing key generation, manual key entry, or key zeroization. To prevent the inadvertent output of sensitive information, two independent internal actions shall be required to output data via any output interface through which plaintext cryptographic keys or CSPs or sensitive data are output (e.g., two different software flags are set, one of which may be user initiated; or two hardware gates are set serially from two separate actions).

Either,

- the physical port(s) used for the input and output of plaintext cryptographic key components, authentication data, and CSPs shall be physically separated from all other ports of the TSF
  or
- the logical interfaces used for the input and output of plaintext cryptographic key components, authentication data, and CSPs shall be logically separated from all other interfaces using a trusted path,
  and
- plaintext cryptographic key components, authentication data, and other CSPs shall be directly entered into the TSF (e.g., via a trusted path or directly attached cable).

ADV_HLD - High-level design

The high-level design of a TOE provides a description of the TSF in terms of major structural units (i.e. subsystems) and 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 TOE security functional requirements.
The high-level design refines the functional specification into subsystems. For each subsystem of the TSF, the high-level design describes its purpose and function, and identifies the security functions contained in the subsystem. The interrelationships of all subsystems are also defined in the high-level design. These interrelationships will be represented as external interfaces for data flow, control flow, etc., as appropriate.

ADV_HLD.3 Semiformal high-level design

ADV_HLD.3.1C The presentation of the high-level design shall be semiformal.

ADV_HLD.3.1D The developer shall provide the high-level design of the TSF. Content and presentation of evidence elements:

ADV_HLD.3.2C The high-level design shall be internally consistent.

ADV_HLD.3.3C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.3.4C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.3.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.3.6C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.3.7C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.3.8C The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing complete details of all effects, exceptions and error messages.

ADV_HLD.3.9C The high-level design shall describe the separation of the TOE into TSP-enforcing and other subsystems.

ADV_IMP - Implementation representation

The description of the implementation representation in the form of source code, firmware, hardware drawings, etc. captures the detailed internal workings of the TSF in support of analysis.

ADV_IMP.3 Structured implementation of the TSF

The ADV_IMP.3.2E element defines a requirement that the evaluator determine that the implementation representation 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 implementation representation, in addition to the pairwise correspondences required by the ADV_RCR family. It is expected that the evaluator will use the evidence provided in ADV_RCR as an input to making this determination.

ADV_IMP.3.1C 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.3.1D The developer shall provide the implementation representation for the entire TSF.

ADV_IMP.3.2C The implementation representation shall be internally consistent.

ADV_IMP.3.3C The implementation representation shall describe the relationships between all portions of the implementation.

ADV_IMP.3.4C The implementation representation shall be structured into small and comprehensible sections.

ADV_INT - TSF internals

This family addresses the internal structure of the TSF. Requirements are presented for modularity, layering (to separate levels of abstraction and minimise circular dependencies), minimisation of the complexity of policy enforcement mechanisms, and the minimisation of the amount of non-TSP-enforcing functionality within the TSF - thus resulting in a TSF that is simple enough to be analysed.
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. Thus, a modular design provides the basis for determining the scope of interaction with other elements of the TSF, provides for increased assurance that unexpected effects do not occur, and also provides the basis for designing and evaluating test suites.

ADV_INT.3 Minimisation of complexity

ADV_INT.3.1C The architectural description shall identify the modules of the TSF and shall specify which portions of the TSF enforce the access control and/or information flow control policies.

ADV_INT.3.1D The developer shall design and structure the TSF in a modular fashion that avoids unnecessary interactions between the modules of the design.

ADV_INT.3.2C The architectural description shall describe the purpose, interface, parameters, and side-effects of each module of the TSF.

ADV_INT.3.2D The developer shall provide an architectural description.

ADV_INT.3.3C The architectural description shall describe how the TSF design provides for largely independent modules that avoid unnecessary interactions.

ADV_INT.3.3D The developer shall design and structure the TSF in a layered fashion that minimises mutual interactions between the layers of the design.

ADV_INT.3.4C The architectural description shall describe the layering architecture.

ADV_INT.3.4D The developer shall design and structure the TSF in such a way that minimises the complexity of the entire TSF.

ADV_INT.3.5C The architectural description shall show that mutual interactions have been minimised, and justify those that remain.

ADV_INT.3.5D The developer shall design and structure the portions of the TSF that enforce any access control and/or information flow control policies such that they are simple enough to be analysed.

ADV_INT.3.6C The architectural description shall describe how the entire TSF has been structured to minimise complexity.

ADV_INT.3.6D The developer shall ensure that functions whose objectives are not relevant for the TSF are excluded from the TSF modules.

ADV_INT.3.7C The architectural description shall justify the inclusion of any non-TSPenforcing modules in the TSF.

ADV_LLD - Low-level design

The low-level design of a TOE provides a description of the internal workings of the TSF in terms of modules and their interrelationships and dependencies. The low-level design provides assurance that the TSF subsystems have been correctly and effectively refined.
For each module of the TSF, the low-level design describes its purpose, function, interfaces, dependencies, and the implementation of any TSP-enforcing functions.

ADV_LLD.1 Descriptive low-level design

ADV_LLD.1.10C The low-level design shall describe the separation of the TOE into TSPenforcing and other modules.

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

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

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

ADV_LLD.1.3C The low-level design shall describe the TSF in terms of modules.

ADV_LLD.1.4C The low-level design shall describe the purpose of each module.

ADV_LLD.1.5C The low-level design shall define the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.

ADV_LLD.1.6C The low-level design shall describe how each TSP-enforcing function is provided.

ADV_LLD.1.7C The low-level design shall identify all interfaces to the modules of the TSF.

ADV_LLD.1.8C The low-level design shall identify which of the interfaces to the modules of the TSF are externally visible.

ADV_LLD.1.9C The low-level design shall describe the purpose and method of use of all interfaces to the modules of the TSF, providing details of effects, exceptions and error messages, as appropriate.

ADV_RCR - Representation correspondence

The correspondence between the various TSF representations (i.e. TOE summary specification, functional specification, high-level design, low-level design, implementation representation) addresses the correct and complete instantiation of the requirements to the least abstract TSF representation provided. This conclusion is achieved by step-wise refinement and the cumulative results of correspondence determinations between all adjacent abstractions of representation.

ADV_RCR.2 Semiformal correspondence demonstration

ADV_RCR.2.1C For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

ADV_RCR.2.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

ADV_RCR.2.2C For each adjacent pair of provided TSF representations, where portions of both representations are at least semiformally specified, the demonstration of correspondence between those portions of the representations shall be semiformal.

ADV_SPM - Security policy modeling

It is the objective of this family to provide additional assurance that the security functions in the functional specification enforce the policies in the TSP. This is accomplished via the development of a security policy model that is based on a subset of the policies of the TSP, and establishing a correspondence between the functional specification, the security policy model, and these policies of the TSP.

ADV_SPM.2 Semiformal TOE security policy model

ADV_SPM.2.1C The TSP model shall be semiformal.

ADV_SPM.2.1D The developer shall provide a TSP model.

ADV_SPM.2.2C The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.2.2D The developer shall demonstrate correspondence between the functional specification and the TSP model.

ADV_SPM.2.3C 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.2.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model.

ADV_SPM.2.5C Where the functional specification is at least semiformal, the demonstration of correspondence between the TSP model and the functional specification shall be semiformal.

5.2.4 AGD - Guidance documents

The guidance documents class provides the requirements for user and administrator guidance documentation. For the secure administration and use of the TOE it is necessary to describe all relevant aspects for the secure application of the TOE.

AGD_ADM - Administrator guidance

Administrator guidance refers to written material that is intended to be used by those persons responsible for configuring, maintaining, and administering the TOE in a correct manner for maximum security. Because the secure operation of the TOE is dependent upon the correct performance of the TSF, persons responsible for performing these functions are trusted by the TSF. Administrator guidance is intended to help administrators understand the security functions provided by the TOE, including both those functions that require the administrator to perform security-critical actions and those functions that provide security-critical information.

AGD_ADM.1 Administrator guidance

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

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

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 behaviour that are relevant to secure operation of the TOE.

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

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

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

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

AGD_USR - User guidance

User guidance refers to material that is intended to be used by non-administrative human users of the TOE, and by others (e.g. programmers) using the TOE's external interfaces. User guidance describes the security functions provided by the TSF and provides instructions and guidelines, including warnings, for its secure use.

AGD_USR.1 User guidance

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

AGD_USR.1.1D The developer shall provide user guidance.

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 behaviour 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.

5.2.5 ALC - Life cycle support

Life-cycle support is an aspect of establishing discipline and control in the processes of refinement of the TOE during its development and maintenance. Confidence in the correspondence between the TOE security requirements and the TOE is greater if security analysis and the production of the evidence are done on a regular basis as an integral part of the development and maintenance activities.

ALC_DVS - Development security

Development security is concerned with physical, procedural, personnel, and other security measures that may be used in the development environment to protect the TOE. It includes the physical security of the development location and any procedures used to select development staff.

ALC_DVS.2 Sufficiency of security measures

ALC_DVS.2.1C 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.2.1D The developer shall produce development security documentation.

ALC_DVS.2.2C The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

ALC_DVS.2.3C The evidence shall justify that the security measures provide the necessary level of protection to maintain the confidentiality and integrity of the TOE.

ALC_FLR - Flaw remediation

Flaw remediation requires that discovered security flaws be tracked and corrected by the developer. Although future compliance with flaw remediation procedures cannot be determined at the time of the TOE evaluation, it is possible to evaluate the policies and procedures that a developer has in place to track and correct flaws, and to distribute the flaw information and corrections.

ALC_FLR.3 Systematic flaw remediation

ALC_FLR.3.1C The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.3.1D The developer shall document the flaw remediation procedures.

ALC_FLR.3.2C 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.3.2D The developer shall establish a procedure for accepting and acting upon user reports of security flaws and requests for corrections to those flaws.

ALC_FLR.3.3C The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.3.3D The developer shall designate one or more specific points of contact for user reports and inquiries about security issues involving the TOE.

ALC_FLR.3.4C 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.3.5C The procedures for processing reported security flaws shall ensure that any reported flaws are corrected and the correction issued to TOE users.

ALC_FLR.3.6C The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.

ALC_FLR.3.7C The flaw remediation procedures shall include a procedure requiring timely responses for the automatic distribution of security flaw reports and the associated corrections to registered users who might be affected by the security flaw.

ALC_LCD - Life cycle definition

Poorly controlled development and maintenance of the TOE can result in a flawed implementation of a TOE (or a TOE that does not meet all of its security requirements). This, in turn, results in security violations. Therefore, it is important that a model for the development and maintenance of a TOE be established as early as possible in the TOE's life-cycle.
Using a model for the development and maintenance of a TOE does not guarantee that the TOE will be free of flaws, nor does it guarantee that the TOE will meet all of its security functional requirements. It is possible that the model chosen will be insufficient or inadequate and therefore no benefits in the quality of the TOE can be observed. Using a life-cycle model that has been approved by some group of experts (e.g. academic experts, standards bodies) improves the chances that the development and maintenance models will contribute to the overall quality of the TOE.

ALC_LCD.3 Measurable life-cycle model

ALC_LCD.3.1C The life-cycle definition documentation shall describe the model used to develop and maintain the TOE, including the details of its arithmetic parameters and/or metrics used to measure the TOE development against the model.

ALC_LCD.3.1D The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

ALC_LCD.3.2C The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.

ALC_LCD.3.2D The developer shall provide life-cycle definition documentation.

ALC_LCD.3.3C The life-cycle definition documentation shall explain why the model was chosen.

ALC_LCD.3.3D The developer shall use a standardised and measurable life-cycle model to develop and maintain the TOE.

ALC_LCD.3.4C The life-cycle definition documentation shall explain how the model is used to develop and maintain the TOE.

ALC_LCD.3.4D The developer shall measure the TOE development using the standardised and measurable life-cycle model.

ALC_LCD.3.5C The life-cycle definition documentation shall demonstrate compliance with the standardised and measurable life-cycle model.

ALC_LCD.3.6C The life-cycle documentation shall provide the results of the measurements of the TOE development using the standardised and measurable life-cycle model.

ALC_TAT - Tools and techniques

Tools and techniques is an aspect of selecting tools that are used to develop, analyse and implement the TOE. It includes requirements to prevent ill-defined, inconsistent or incorrect development tools from being used to develop the TOE. This includes, but is not limited to, programming languages, documentation, implementation standards, and other parts of the TOE such as supporting runtime libraries.

ALC_TAT.2 Compliance with implementation standards

ALC_TAT.2.1C All development tools used for implementation shall be well-defined.

ALC_TAT.2.2C The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

ALC_TAT.2.3C The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.

ALC_TAT.2.3D The developer shall describe the implementation standards to be applied.

5.2.6 ATE - Tests

The class "Tests" encompasses four families: coverage (ATE_COV), depth (ATE_DPT), independent testing (e.g. functional testing performed by evaluators) (ATE_IND), and functional tests (ATE_FUN). Testing helps to establish that the TOE security functional requirements are met. Testing provides assurance that the TOE satisfies at least the TOE security functional requirements, although it cannot establish that the TOE does no more than what was specified. Testing may also be directed toward the internal structure of the TSF, such as the testing of subsystems and modules against their specifications.

ATE_COV - Coverage

This family addresses those aspects of testing that deal with completeness of test coverage. That is, it addresses the extent to which the TSF is tested, and whether or not the testing is sufficiently extensive to demonstrate that the TSF operates as specified.

ATE_COV.2 Analysis of coverage

In this component, the objective is to establish that the TSF has been tested against its functional specification in a systematic manner. This is to be achieved through an examination of developer analysis of correspondence.
The developer is required to demonstrate that the tests which have been identified include testing of all of the security functions as described in the functional specification. The analysis should not only show the correspondence between tests and security functions, but should provide also sufficient information for the evaluator to determine how

the functions have been exercised. This information can be used in planning for additional evaluator tests. Although at this level the developer has to demonstrate that each of the functions within the functional specification has been tested, the amount of testing of each function need not be exhaustive.

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

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.

ATE_DPT - Depth

The components in this family deal with the level of detail to which the TSF is tested. Testing of security functions is based upon increasing depth of information derived from analysis of the representations.
The objective is to counter the risk of missing an error in the development of the TOE. Additionally, the components of this family, especially as testing is more concerned with the internal structure of the TSF, are more likely to discover any malicious code that has been inserted.
Testing that exercises specific internal interfaces can provide assurance not only that the TSF exhibits the desired external security behaviour, but also that this behaviour stems from correctly operating internal mechanisms.

ATE_DPT.3 Testing: implementation representation

The subsystems of a TSF provide a high-level description of the internal workings of the TSF. Testing at the level of the subsystems, in order to demonstrate the presence of any flaws, provides assurance that the TSF subsystems have been correctly realised.
The modules of a TSF provide a description of the internal workings of the TSF. Testing at the level of the modules, in order to demonstrate the presence of any flaws, provides assurance that the TSF modules have been correctly realised.
The implementation representation of a TSF provides a detailed description of the internal workings of the TSF. Testing at the level of the implementation, in order to demonstrate the presence of any flaws, provides assurance that the TSF implementation has been correctly realised.
The developer is expected to describe the testing of the high-level design of the TSF in terms of "subsystems". The term "subsystem" is used to express the notion of decomposing the TSF into a relatively small number of parts.
The developer is expected to describe the testing of the low-level design of the TSF in terms of "modules". The term "modules" is used to express the notion of decomposing each of the "subsystems" of the TSF into a relatively small number of parts.
The implementation representation is the one which is used to generate the TSF itself (e.g. source code which is then compiled).

ATE_DPT.3.1C 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, low-level design and implementation representation.

ATE_DPT.3.1D The developer shall provide the analysis of the depth of testing.

ATE_FUN - Functional tests

Functional testing performed by the developer establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of its PP/ST. Such functional testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. The family "Functional tests" is focused on the type and amount of documentation or support tools required, and what is to be demonstrated through developer testing. Functional testing is not limited to positive confirmation that the required security functions are provided, but may also include negative testing to check for the absence of particular undesired behaviour (often based on the inversion of functional requirements).

ATE_FUN.2 Ordered functional testing

The objective is for the developer to demonstrate that all security functions perform as specified. The developer is required to perform testing and to provide test documentation. In this component, an additional objective is to ensure that testing is structured such as to avoid circular arguments about the correctness of the portions of the TSF being tested.

ATE_FUN.2.1C The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.

ATE_FUN.2.1D The developer shall test the TSF and document the results.

ATE_FUN.2.2C The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.2.2D The developer shall provide test documentation.

ATE_FUN.2.3C 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.2.4C The expected test results shall show the anticipated outputs from a successful execution of the tests.

ATE_FUN.2.5C The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.

ATE_FUN.2.6C The test documentation shall include an analysis of the test procedure ordering dependencies.

ATE_IND - Independent testing

One objective is to demonstrate that the security functions perform as specified.
An additional objective is to counter the risk of an incorrect assessment of the test outcomes on the part of the developer that results in the incorrect implementation of the specifications, or overlooks code that is non-compliant with the specifications.

ATE_IND.2 Independent testing - sample

The objective is to demonstrate that the security functions perform as specified. Evaluator testing includes selecting and repeating a sample of the developer tests.
The intent is that the developer should provide the evaluator with materials necessary for the efficient reproduction of developer tests. This may include such things as machine-readable test documentation, test programs, etc.
This component contains a requirement that the evaluator has available test results from the developer to supplement the programme of testing. The evaluator will repeat a sample of the developer's tests to gain confidence in the results obtained. Having established such confidence the evaluator will build upon the developer's testing by conducting additional tests that exercise the TOE in a different manner. By using a platform of validated developer test results the evaluator is able to gain confidence that the TOE operates correctly in a wider range of conditions than would be possible purely using the developer's own efforts, given a fixed level of resource. Having gained confidence that the developer has

tested the TOE, the evaluator will also have more freedom, where appropriate, to concentrate testing in areas where examination of documentation or specialist knowledge has raised particular concerns.

ATE_IND.2.1C The TOE shall be suitable for testing.

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

ATE_IND.2.2C The developer shall provide an equivalent set of resources to those that were used in the developerŠs functional testing of the TSF.

5.2.7 AVA - Vulnerability assessment

The class addresses the existence of exploitable covert channels, the possibility of misuse or incorrect configuration of the TOE, the possibility to defeat probabilistic or permutational mechanisms, and the possibility of exploitable vulnerabilities introduced in the development or the operation of the TOE.

AVA_CCA - Covert channel analysis

Covert channel analysis is carried out to determine the existence and potential capacity of unintended signalling channels (i.e. illicit information flows) that may be exploited.
The assurance requirements address the threat that unintended and exploitable signalling paths exist that may be exercised to violate the SFP.

AVA_CCA.1 Covert channel analysis

The objective is to identify covert channels that are identifiable, through an informal search for covert channels.

AVA_CCA.1.1C The analysis documentation shall identify covert channels and estimate their capacity.

AVA_CCA.1.1D The developer shall conduct a search for covert channels for each information flow control policy.

AVA_CCA.1.2C The analysis documentation shall describe the procedures used for determining the existence of covert channels, and the information needed to carry out the covert channel analysis.

AVA_CCA.1.2D The developer shall provide covert channel analysis documentation.

AVA_CCA.1.3C The analysis documentation shall describe all assumptions made during the covert channel analysis.

AVA_CCA.1.4C The analysis documentation shall describe the method used for estimating channel capacity, based on worst case scenarios.

AVA_CCA.1.5C The analysis documentation shall describe the worst case exploitation scenario for each identified covert channel.

AVA_MSU - Misuse

Misuse investigates whether the TOE can be configured or used in a manner that is insecure but that an administrator or user of the TOE would reasonably believe to be secure.

AVA_MSU.2 Validation of analysis

The objective is to ensure that misleading, unreasonable and conflicting guidance is absent from the guidance documentation, and that secure procedures for all modes of operation have been addressed. Insecure states should be easy to detect. In this component, an analysis of the guidance documentation by the developer is required to provide additional assurance that the objective has been met.

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.2.1D The developer shall provide guidance documentation.

AVA_MSU.2.2C The guidance documentation shall be complete, clear, consistent and reasonable.

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

AVA_MSU.2.3C The guidance documentation shall list all assumptions about the intended environment.

AVA_MSU.2.4C The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).

AVA_MSU.2.5C The analysis documentation shall demonstrate that the guidance documentation is complete.

AVA_SOF - Strength of TOE security functions

Even if a TOE security function cannot be bypassed, deactivated, or corrupted, it may still be possible to defeat it because there is a vulnerability in the concept of its underlying security mechanisms. For those functions a qualification of their security behaviour can be made using the results of a quantitative or statistical analysis of the security behaviour of these mechanisms and the effort required to overcome them. The qualification is made in the form of a strength of TOE security function claim.

AVA_SOF.1 Strength of TOE security function evaluation

Security functions are implemented by security mechanisms. For example, a password mechanism can be used in the implementation of the identification and authentication security function.
The strength of TOE security function evaluation is performed at the level of the security mechanism, but its results provide knowledge about the ability of the related security function to counter the identified threats.
The strength of TOE security function analysis should consider at least the contents of all the TOE deliverables, including the ST, for the targeted evaluation assurance level.

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

AVA_SOF.1.2C For each mechanism with a specific strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the specific strength of function metric defined in the PP/ST.

AVA_VLA - Vulnerability analysis

Vulnerability analysis is an assessment to determine whether vulnerabilities identified, during the evaluation of the construction and anticipated operation of the TOE or by other methods (e.g. by flaw hypotheses), could allow users to violate the TSP.

AVA_VLA.2 Independent vulnerability analysis

A vulnerability analysis is performed by the developer to ascertain the presence of security vulnerabilities, and to confirm that they cannot be exploited in the intended environment for the TOE.

The evaluator performs independent penetration testing, supported by the evaluator's independent vulnerability analysis, to determine that the TOE is resistant to penetration attacks performed by attackers possessing a low attack potential.

AVA_VLA.2.1C The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.2.1D The developer shall perform and document an analysis of the TOE deliverables searching for ways in which a user can violate the TSP.

AVA_VLA.2.2C The documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks.

AVA_VLA.2.2D The developer shall document the disposition of identified vulnerabilities.

Chapter 6

PP APPLICATION NOTES

6.1 Notes on evaluating this PP

6.1.1 Document construction method

By construction, this PP can be shown to be structurally correct. See the tool
`http://www.safelayer.com/download/pkipp/7.PPhelper`
and the knowledge database [NSA00], to gain an idea of its development method.

By supporting the PP writing with a tool, PP revision and definition is centered on conceptual discussion based in the supporting knowledge database. Validation of this document rationale is equivalent to validation of [NSA00].

6.1.2 Concepts mapping rationale and deviations from "The Common Criteria Profiling Knowledge Base".

Figure 7.1 shows the conceptual diagram used in this document. This diagram is initially derived from "Figure 4.5 - Derivation of requirements and specifications" at [the99b], with some extensions:

- Threats are decomposed into Detailed Attacks. Whereas Security Objectives are directly mapped at the [the99a] to Threats, in our model, Security Objectives are directly mapped to Detailed Attacks, and in turn, to Threats.
- The same decomposition is applied to Security Policies, where in our model there is a General Security Policy Statements level, and a further Detailed Security Policy Statements who are directly mapped to Security Objectives.

  These two extensions have been inherited from the [NSA00] model. A further one complements this [NSA00]:
- Assumptions are mapped to Threats and Policies not completely satisfied by the TOE, and in particular, to those Security Objectives not satisfied solely by the TOE. This mapping is missing from [NSA00], where assumptions are not directly linked to Security Objectives.

Actual mappings between these concepts are embedded into the PPhelper tools, as imported from [NSA00]. The source of [NSA00] is by itself an indication of correctness of the mappings. These have been further assessed by the PP "*PKI PP working group*" to gain definitive confidence in their applicability and correctness with highly positive returns. Some minor details were detected and corrected directly in the PPhelper Knowledge Base, mostly due to component leveling that did not affect the validity of the mappings.

Two general items included in [NSA00] have been eliminated in our final PPhelper Knowledge Base.

1. Some functional extensions regarding TOE hardware protection have been removed, since no hardware security claims are intended to be included in this PP. This is to allow both software and hardware implementations of the functionality mandated herein, so that in the latter case hardware security requirements can be added in the TOE ST as required.
2. A number of threats and attacks regarding TOE protection against cryptanalysis have also been removed as considered not applicable. This is justified by the fact that PKIs precisely produce pairs of plain text and their correspondent ciphered text, in the form of certificates. The fact that this may put the TOE under risk is assumed, and even quantified in the Information Flow Model and Policy. (see 5.1.4).

6.1.3 Inclusion of functional and assurance requirements from "FIPS 140-2 SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES"

As already stated, requirements emanating from [fip] have been incorporated into this document where there are requirements in both this PP and [fip] that require similar functionality. This has been done to align it with a current market recognized authority for cryptographic module security standard, so that a TOE complying with this PP does not have to define similar but different functionality to achieve compliance of [fip] for the core PSK cryptographic module.

Table 6.1.3 shows the correspondence between the [fip] original requirement, and the PP equivalence. Most of these imported requirements have been included in verbatim, whereas some minor adjustments have been performed in other cases.

FIPS 140 Security Requirement	PP requirement
4.2 Cryptographic Module Ports and Interfaces	ADV_FSP.3.7C TSF Ports and Interfaces. See 5.2.3
4.3.1 Roles	FDP_ACC.2 Complete Access Control . See 5.1.4, and FMT_SMR.2 Restrictions on security roles. See 5.1.6
4.4 Finite State Model	ADV_FSP.3.6C Finite State Model. See 5.2.3
4.7 Cryptographic Key Management	FCS Cryptographic support. See 5.1.3.
4.9 Self-Test	FPT_TST TSF self test. See 5.1.7

Table 6.1: FIPS 140-2 mapping to PP requirements.

6.2 Notes on using this PP

The PSK concept allows for multiple design solutions, as a stand-alone product, or as a secure crypto management core, and in all those cases, compliance can be claimed with respect to this PP.

Software-only solutions may find in this PP a self-contained specification that will drive a product with a very high security profile.

Solutions incorporating hardware tokens or elements may be tempted to relax the environmental requirements by introducing security requirements regarding hardware. This could be in the form of emission control, tamper active reaction, etc. If properly selected, all the environmental assumptions can be eliminated.

Since this PP includes most of the non-hardware requirements of [fip], as detailed in Table 6.1.3, TOEs should be validated against this standard without requiring modification of their functionality. In particular, compliance with respect to non-hardware requirements of [fip] gets up to level 4, provided that the TOE assurance component ADV_SPM is raised to formal level.

Chapter 7

RATIONALE

This PP is based upon the concepts and relationships shown in Figure 7.1. The Security Environment, already defined, states the identified General Threats, General Organizational Policies and General Assumptions.

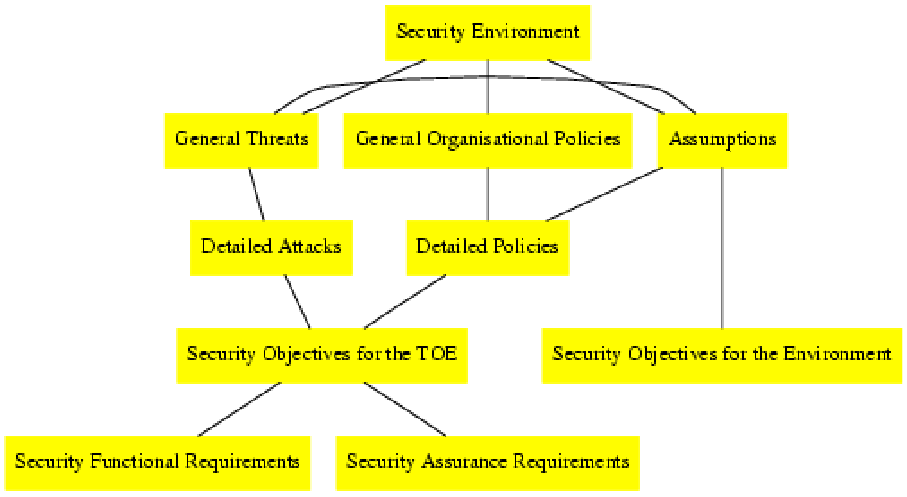
Security Environment
General Threats
General Organisational Policies
Assumptions
Detailed Attacks
Detailed Policies
Security Objectives for the TOE
Security Objectives for the Environment
Security Functional Requirements
Security Assurance Requirements

Figure 7.1: Concepts and relationships

A General Threat can be considered as grouping a number of different but related Detailed Attacks. While the General Threat is the abstraction of the threat class, Detailed Attacks are the members of the class. The TOE must be able to counter the identified Detailed Attacks for it to state that it counters the General Threat.In this rationale, Detailed Attacks that compose each General Threat are analyzed, and for each one of these attacks, particular Security Objectives are determined.

General Policies are again exploded into more practical, less abstract, Detailed Policies, whose sum is equivalent to the General Policy Statement. The General Policy is not the abstraction of a policy class, but divided into Detailed Policies. Security Objectives are identified to implement each of these Detailed Policies, and in sum, the General Policy Statement.

Assumptions reflect both threat assumptions (AT.) and policy assumptions (AP.)

Assumptions regarding General Threats can be read with a double meaning, either as if the environment where responsible for countering with the threat, with no residual risk, or as if the risk of suffering the threat is accepted. Since General Threats are related to a group of Detailed Attacks, usually a TOE may address some of these attacks, but not all of them so as to claim the General Threat countering. In this case, the assumption has been limited to those attacks not addressed directly by the TOE, but in collaboration with the environment, the General Threat is countered completely.

Assumptions regarding Policies are stated at Detailed Policy Statements level, in the understanding that General Policy Statements must always be applied to the TOE, otherwise being irrelevant to this PP. Assumptions about policies are read as if the environment is responsible for fulfilling the particular assigned Detailed Policy Statement, where combined with the TOE, full General Policy Statement coverage is obtained.

All assumptions are finally analyzed to determine the Security Objectives that should be mapped to the environment.

Those Security Objectives that are relevant to the TOE are then analyzed to determine the required TOE Security Requirements, both Functional and Assurance.

7.1 General Threats and Attacks rationale.

As already explained, General Threats identified in Sec. 3.3 and Sec. 3.5 represent categories of detailed attacks. A mapping of coverage is presented in this section, where the assigned attacks to each Threat are shown, ensuring that no threat is left void of attacks, and that every threat is assigned at least to a General Threat class. Note that in some cases, a particular Detailed Attacks may be considered appropriate to be present in more than one General Threat Category.

7.1.1 Administrative errors of commission

T.Admin_Err_Commit (see 3.3.1).

The following Detailed Attack(s) form this General Threat category:

DA.Adm_Err_Crypto (see 3.4.1). Accidental mismanagement of cryptographic functions

An administrator misconfigures cryptographic functions or stores plaintext keys in insecure areas.

This threat is countered solely by the TOE.

DA.Admin_Err_AC_Policy (see 3.4.11). Administrator error modifies access control or information flow policy

An administrator's error in data entry changes the access control or information flow policy enforced by the system in such a way that it no longer serves its intended purpose.

This threat is countered solely by the TOE.

DA.Admin_Err_Audit (see 3.4.12). Administrator error changes audit behavior

An administrator's error in data entry changes the audit behavior of the system in such a way that auditing no longer serves its intended purpose.

This threat is countered solely by the TOE.

DA.Admin_Err_Authentic (see 3.4.13). Administrator error modifies authentication enforcement

An administrator's error in data entry changes the authentication-enforcement mechanism of the system in such a way that it no longer serves its intended purpose.

This threat is countered solely by the TOE.

DA.Admin_Err_Info (see 3.4.14). Administrator error makes information unavailable

An administrator's error in data entry makes system or application information unavailable.

This threat is countered solely by the TOE.

DA.Admin_Err_Resource (see 3.4.16). Administrator error makes resource unavailable

An administrator's error in data entry makes system or application resources unavailable.

This threat is countered solely by the TOE.

DA.Admin_Err_Sys_Entry (see 3.4.17). Administrator error modifies entry policy

An administrator's error in data entry changes the intended entry policy of the system or application.

This threat is countered solely by the TOE.

DA.Admin_Err_User_Attr (see 3.4.19). Administrator error modifies user security attributes

An administrator's error in data entry modifies a user's security attributes, which makes the attributes inappropriate under the security policy of the system or application.

This threat is countered solely by the TOE.

7.1.2 Administrative errors of omission

T.Admin_Err_Omit (see 3.3.2).
The following Detailed Attack(s) form this General Threat category:

DA.Adm_Err_Crypto (see 3.4.1). Accidental mismanagement of cryptographic functions

An administrator misconfigures cryptographic functions or stores plaintext keys in insecure areas.

This threat is countered solely by the TOE.

DA.Adm_Misconfig_User (see 3.4.10). User privileges and/or authorizations are not updated upon reassignment

A change in the status of users duties do not get reflected in administratively controlled privileges and/or authorizations.

This threat is countered solely by the TOE.

DA.Admin_Err_Omit_Trap (see 3.4.15). Back door left open

An administrator inadvertently leaves a back door port open after routine maintenance, allowing continuing unauthorized access by the service organization.

This threat is countered solely by the TOE.

DA.Admin_Err_Update (see 3.4.18). Administrator fails to update security configuration

The organizational security policies changes but these changes are not reflected in all system configurations, resulting in circumvention and/or incorrect application of security policies.

This threat is countered solely by the TOE.

7.1.3 Hostile administrator modification of user or system data

T.Admin_Hostile_Modify (see 3.3.3).
The following Detailed Attack(s) form this General Threat category:

DA.Adm_Hstl_Audit_Dstr (see 3.4.2). Destruction or modification of audit data

An administrator seeks to cover up misbehavior by destroying and/or falsifying audit data.

This threat is countered solely by the TOE.

DA.Adm_Hstl_Mod_Data_AC (see 3.4.4). Administrator maliciously modifies or deletes data access control attributes

An administrator maliciously modifies access control attributes, allowing the administrator or other perpetrator to gain access and manipulative capability to organizational assets, contrary to organizational policy.

This threat is countered solely by the TOE.

DA.Adm_Hstl_Mod_DataAps (see 3.4.3). Administrator modifies or destroys user data or applications

The administrator abuses IT or user trust, as being the administrator and without changing the user imposed data security attributes, by destroying data or applications for malicious reasons or to cover up misappropriate behavior.

This threat is countered solely by the TOE.

DA.Adm_Hstl_Mod_IFC (see 3.4.5). The administrator maliciously modifies information flow control.

The administrator maliciously alters information flow control policy to allow information to flow to inappropriate locations for unauthorized users access or modification.

This threat is countered solely by the TOE.

DA.Adm_Hstl_Mod_SEP (see 3.4.6). Administrator maliciously modifies system entry parameters

An administrator or user masquerading as an administrator maliciously modifies system entry parameters which would allow unauthorized access to an organization's protected assets.

This threat is countered solely by the TOE.

DA.Adm_Hstl_Mod_TSFCode (see 3.4.7). Administrator maliciously modifies security-critical code

The administrator modifies the security-critical (TSF) code to weaken the security effectiveness of the TSF or introduce a new security breech.

This threat is countered solely by the TOE.

DA.Adm_Hstl_Mod_USB (see 3.4.8). Administrator maliciously modifies user/subject bindings

The administrator modifies a user/subject binding which would allow a user to act on an object without creating an audit trail.

This threat is countered solely by the TOE.

DA.Adm_Hstl_Mod_UsrAttr (see 3.4.9). Administrator maliciously modifies user attributes and/or roles

The administrator modifies or mishandles the users attributes or roles which allows users, unauthorized or authorized, to have the ability to perform inappropriate actions or could prevent a user from performing an authorized action.

This threat is countered solely by the TOE.

7.1.4 Software containing security-related flaws

T.Dev_Flawed_Code (see 3.3.4).

The following Detailed Attack(s) form this General Threat category:

DA.Dev_FC_Attr_Interp (see 3.4.20). Inconsistent interpretation of audit data attributes

The security-critical (TSF) components inconsistently interpret audit data attributes exchanged with another trusted IT product.

This threat is countered solely by the TOE.

DA.Dev_FC_Buff_Not_Clr (see 3.4.21). Buffers not cleared by the system

The system leaves user information in a system buffer for view by another unauthorized user.

This threat is countered solely by the TOE.

DA.Dev_FC_Ctrl_Data (see 3.4.22). Incorrect modification of control data

A security-critical (TSF) component incorrectly modifies control data regarding a user process.

This threat is countered solely by the TOE.

DA.Dev_FC_Data_Export (see 3.4.23). System data incorrectly exchanged

The system incorrectly exchanges system data with another trusted system.

This threat is countered solely by the TOE.

DA.Dev_FC_Recovery (see 3.4.24). Non-secure recovery

A system failure may alter the behavior of the system's security functions in such a way that, upon recovery, it no longer properly enforces its security policy (TSP).

This threat is countered solely by the TOE.

DA.Dev_FC_Replication (see 3.4.25). Inaccurate system-data replication

The system does not accurately replicate system data to different parts of the system where replication is required.

This threat is countered solely by the TOE.

DA.Dev_FC_Self_Protect (see 3.4.26). System modification by unauthorized source

Software developer or hacker modifies system security functions resulting in a loss of security protection.

This threat is countered solely by the TOE.

DA.Dev_FC_Trap_Door (see 3.4.27). Malicious developer creates secret trapdoor in system

The system developer creates a secret back door in the system (TOE) that allows covert access by the developer. This allows the developer to collect information, monitor user actions, modify the operation of the TOE, or just make unauthorized use of the TOE.

This threat is countered solely by the TOE.

7.1.5 Hacker undetected system access

T.Hack_AC (see 3.3.5).

The following Detailed Attack(s) form this General Threat category:

DA.Hack_AC_Code_Vul (see 3.4.28). Hacker gains access through a vulnerability in code

The hacker can use vulnerabilities found in system or application code to break into a system undetected.

This threat is countered solely by the TOE.

DA.Hack_AC_Weak (see 3.4.29). Weak system access control mechanism or system access control implementation

The system access control mechanism(s) or user attributes are weak and can be broken or the implementation methods of the system access control causes the weakness.

This threat is countered solely by the TOE.

7.1.6 Message content modification

T.Hack_Msg_Data (see 3.3.6).

The following Detailed Attack(s) form this General Threat category:

DA.Hack_MsgData_RcvTSF (see 3.4.36). Modification of security-critical data in transit from a remote trusted site

Security-critical (TSF) data is modified in transit from a remote trusted site, either accidentally by the communications infrastructure or deliberately by a hostile outsider.

This threat is countered solely by the TOE.

DA.Hack_MsgData_RcvUsr (see 3.4.37). Modification of user data in transit from a remote site

A hostile outsider modifies message data in route to the system. Alternatively, errors in the communications infrastructure modify the message.

This threat is countered solely by the TOE.

DA.Hack_MsgData_SndTSF (see 3.4.38). Modification of security-critical data in transit to a remote site

Security-critical (TSF) data is modified in transit to a remote site, either accidentally by the communications infrastructure or deliberately by a hostile outsider.

This threat is countered solely by the TOE.

DA.Hack_MsgData_SndUsr (see 3.4.39). Modification of user data in transit to a remote site

A hostile outsider modifies message data in route to a remote site. Alternatively, errors in the communications infrastructure modify the message.

This threat is countered solely by the TOE.

7.1.7 Unexpected disruption of system or component power

T.Power_Disrupt (see 3.3.7).
The following Detailed Attack(s) form this General Threat category:

DA.Power_Disrupt_Reset (see 3.4.43). Unexpected power reset

An unintentional, malicious, or environmentally caused power reset occurs, resulting in the loss of critical information or the system to enter a non-secure state.

This threat is countered solely by the TOE.

7.1.8 Sender denies sending information

T.Repudiate_Send (see 3.3.8).
The following Detailed Attack(s) form this General Threat category:

DA.Repudiate_Send_Int (see 3.4.44). Denial of having sent information to another local user

A local, authorized user sends a message to another local user via the system, and then denies having done it. This affects the recipient of the message as well as any resources allocated or modified by the recipient in response to the message.

This threat is countered solely by the TOE.

DA.Repudiate_Send_Local (see 3.4.45). Denial of having sent information to a remote user

A local, authorized user sends a message to another user at a remote trusted product, and then denies having done it. This affects the recipient of the message as well as any resources allocated or modified by the recipient in response to the message.

This threat is countered solely by the TOE.

DA.Repudiate_Send_Rem (see 3.4.46). Denial of having sent data by a remote user

A local, authorized user receives a message from another user at a remote trusted product who then denies having sent it. This affects the recipient of the message as well as any resources allocated or modified by the recipient in response to the message.

This threat is countered solely by the TOE.

7.1.9 Hostile user acts cause confidentiality breaches

T.User_Abuse_Conf (see 3.3.9).
The following Detailed Attack(s) form this General Threat category:

DA.User_Abuse_Conf_Disk (see 3.4.48). User smuggles data using removable media

A user collects sensitive or proprietary information and improperly removes it from the system by putting it on removable media.

This threat is countered solely by the TOE.

7.1.10 User abuses authorization to collect data

T.User_Collect (see 3.3.10).
The following Detailed Attack(s) form this General Threat category:

DA.User_Collect_Browse (see 3.4.50). User collects data by browsing

An authorized user abuses granted authorizations by browsing files in order to collect data.

This threat is countered solely by the TOE.

DA.User_Collect_Deceive (see 3.4.51). User collects authentication data by deception

An authorized user steals authentication data by emulating a login procedure on an active terminal.

This threat is countered solely by the TOE.

DA.User_Collect_Deduce (see 3.4.52). User collects data by deduction

An authorized user abuses granted authorizations by repeatedly accessing aggregate data in order to deduce specific, sensitive data.

This threat is countered solely by the TOE.

DA.User_Collect_Eaves (see 3.4.53). User collects data by eavesdropping

An authorized user abuses granted authorizations by eavesdropping on communication lines in order to collect data.

This threat is countered solely by the TOE.

DA.User_Collect_Residue (see 3.4.54). User collects residual data

An authorized user collects residual data from public objects after prior usage.

This threat is countered solely by the TOE.

7.1.11 User errors cause confidentiality breaches

T.User_Err_Conf (see 3.3.11).
The following Detailed Attack(s) form this General Threat category:

DA.Hack_Ext_CryptoAsset (see 3.4.33). Accidental or deliberate mishandling of cryptographic assets external to the TOE

Cryptographic assets are mishandled after the leave the TOE, either in transit or while residing on stored media.

This threat is countered solely by the TOE.

DA.User_Err_Conf_Class (see 3.4.58). Under-classification of data sensitivity on export

An authorized user presents confidential or classified information to a recipient, indicating that it is less sensitive than it really is, thereby encouraging the recipient to pass it along to other potentially inappropriate recipients.

This threat is countered solely by the TOE.

DA.User_Err_Conf_Crypto (see 3.4.59). Accidental release of cryptographic assets due to user error

User error causes release of cryptographic assets to unauthorized recipients.

This threat is countered solely by the TOE.

DA.User_Err_Conf_Exp (see 3.4.60). Confidentiality violation of export control policy

An authorized user exposes or exports data in violation of export control policy. The data may be private or classified, the recipient is not authorized to receive it.

This threat is countered solely by the TOE.

7.1.12 User errors cause integrity breaches

T.User_Err_Integrity (see 3.3.12).

The following Detailed Attack(s) form this General Threat category:

DA.Hack_Ext_CryptoAsset (see 3.4.33). Accidental or deliberate mishandling of cryptographic assets external to the TOE

Cryptographic assets are mishandled after the leave the TOE, either in transit or while residing on stored media.

This threat is countered solely by the TOE.

DA.User_Err_AttrXpt (see 3.4.57). Falsification of information quality in data export

An authorized user presents incorrect information, indicating to the recipient that it is correct, thereby encouraging the recipient to make unwarranted use of the information.

This threat is countered solely by the TOE.

DA.User_Modify_Data (see 3.4.67). User improperly modifies user data

An authorized user modifies or deletes user data in violation of organizational policy.

This threat is countered solely by the TOE.

7.1.13 User errors undermine the system's security features

T.User_Err_Slf_Protect (see 3.3.13).

The following Detailed Attack(s) form this General Threat category:

DA.User_Err_MsngAttrXpt (see 3.4.62). Failure to provide object security attributes in data export

An authorized user deliberately or accidentally exports data so that the data is not accompanied by required handling information.

This threat is countered solely by the TOE.

DA.User_Err_Object_Attr (see 3.4.63). Incorrectly set object attributes

An authorized user sets an object's security attributes inappropriately, misdirecting its use. The misdirection may allow unauthorized reading or modification, or it may prohibit authorized reading or modification.

This threat is countered solely by the TOE.

7.1.14 User abuses authorization to modify data

T.User_Modify (see 3.3.14).

The following Detailed Attack(s) form this General Threat category:

DA.User_Modify_Audit (see 3.4.65). User modifies audit trail

An authorized user modifies audit data or audit attributes to avoid accountability.

This threat is countered solely by the TOE.

DA.User_Modify_Auth (see 3.4.66). User improperly modifies authentication data

An authorized user changes the authentication data of another user without first masquerading as that user, in a manner that is not consistent with organizational security policy.

This threat is countered solely by the TOE.

DA.User_Modify_Data (see 3.4.67). User improperly modifies user data

An authorized user modifies or deletes user data in violation of organizational policy.

This threat is countered solely by the TOE.

DA.User_Modify_TSFData (see 3.4.68). User improperly modifies TSF data

User modifies or deletes TSF data undermining security protection.

This threat is countered solely by the TOE.

7.1.15 User abuses authorization to send data

T.User_Send (see 3.3.15).
The following Detailed Attack(s) form this General Threat category:

DA.User_Abuse_Conf_Steg (see 3.4.49). Steganographic data smuggling

An authorized user hides sensitive information in an innocuous-appearing file, for the purpose of covertly passing it to an unauthorized party. The hidden data is undetectable to anyone using the file for its intended purpose, but can be recovered using special techniques.

This threat is countered solely by the TOE.

DA.User_Send_Conf (see 3.4.70). User sends data violating confidentiality

An authorized user abuses granted authorizations and violates export control policy by sending data to a recipient who is not authorized to receive the data.

This threat is countered solely by the TOE.

DA.User_Send_Integrity (see 3.4.71). User sends data violating integrity

An authorized user deliberately exports data inappropriately, with the result that there is a lack of required quality control on the exported data.

This threat is countered solely by the TOE.

7.1.16 Administrator violates user privacy policy

T.Admin_UserPriv (see 3.5.1).
The following Detailed Attacks form this General Threat category:

DA.Admin_UserPriv_Agg (see 3.6.1). Administrator aggregates privacy information

An administrator aggregates information that indirectly reveals the identity (or other privacy related information) of user(s) in violation of user privacy policy.

This attack is to be countered with support from the Environment.

DA.Admin_UserPriv_Col (see 3.6.2). Administrator reads collected user privacy information

An administrator reads information collected by the IT system or product that reveals the identity (or other privacy related information) of user(s) in violation of user privacy policy.

This attack is to be countered with support from the Environment.

DA.Admin_UserPriv_Gen (see 3.6.3). Administrator reads system generated privacy information

An administrator reads information generated by the IT system or product that directly reveals the identity (or other privacy related information) of user(s) in violation of user privacy policy.

This attack is to be countered with support from the Environment.

7.1.17 Malicious code exploitation

T.Malicious_Code (see 3.5.2).
The following Detailed Attacks form this General Threat category:

DA.Mal_Code_Hack_Downld (see 3.6.4). Malicious code perpetrator dissemination

A perpetrator disseminates malicious code via push or pull mechanism.

This attack is to be countered with support from the Environment.

DA.Mal_Code_Hack_Exe (see 3.6.5). Malicious code perpetrator execution

A perpetrator executes malicious code either remotely or locally.

This attack is to be countered with support from the Environment.

DA.Mal_Code_IT_Download (see 3.6.6). Malicious code accidental IT download

An IT device accidentally transfers or downloads malicious code to itself or other device that it can influence.

This attack is to be countered with support from the Environment.

DA.Mal_Code_IT_Exe (see 3.6.7). Malicious code IT execution

An IT device under normal operations enters a state required to execute the malicious code.

This attack is to be countered with support from the Environment.

DA.Mal_Code_Usr_Downld (see 3.6.8). Malicious code accidental user download

An authorized user accidentally downloads malicious code.

This attack is to be countered with support from the Environment.

DA.Mal_Code_Usr_Exe (see 3.6.9). Malicious code user execution

An authorized user executes malicious code accidentally.

This attack is to be countered with support from the Environment.

7.1.18 Legitimate system services are spoofed

T.Spoofing (see 3.5.3).

The following Detailed Attacks form this General Threat category:

DA.Hack_Spoof_Login (see 3.6.10). Login program replicated to capture authentication data

An attacker simulates the system's login program and runs it at an open terminal or workstation in order to capture a legitimate user's authentication data.

This attack is to be countered with support from the Environment.

DA.Hack_Spoof_MsgHdr (see 3.4.40). Attacker modifies protocol headers

An attacker may modify protocol headers such that a user believes the communication is coming from a source that is different from where it was actually sent.

This attack is countered solely by the TOE.

7.1.19 Hacker attempts resource denial of service

T.Hack_Avl_Resource (see 3.5.4).

The following Detailed Attacks form this General Threat category:

DA.Hack_Comm_Overload (see 3.4.32). Hacker causes overload of communication resources

The unauthorized use of communication resources by a hacker causes a denial or delay in service to legitimate operations within the TOE scope of control. This would include the excess bandwidth utilization, leading to the TOE's inability to perform it's security functions.

This attack is countered solely by the TOE.

DA.Hack_Prcsr_Overload (see 3.6.11). Hacker causes system task overload resulting in denial of service

Hacker causes system task overload resulting in denial of service. The system (TOE) has been over-tasked and can not complete the assigned tasking at all or in an expected amount of time. The hacker invokes processing functions in association with unauthorized activity that leads to overburdening processing resources on the TOE.

This attack is to be countered with support from the Environment.

DA.Hack_Stg_Overload (see 3.4.41). Hacker activities cause storage overload

A hacker initiates processes that tax the amount of storage available in the system (TOE). Such would be the case when a hacker floods the TOE with e-mails.

This attack is countered solely by the TOE.

7.1.20 Hacker eavesdrops on user data communications

T.Hack_Comm_Eavesdrop (see 3.5.5).
The following Detailed Attacks form this General Threat category:

DA.Hack_CommEaves_Eman (see 3.4.30). The communication mechanism emanates data
An outsider uses special equipment to capture emanations off the communications line.
This attack is countered solely by the TOE.

DA.Hack_CommEaves_Intrc (see 3.4.31). Outsider intercepts user communications
An outsider who is not an intended recipient intercepts user data communications.
This attack is countered solely by the TOE.

DA.Hack_CommEaves_Tap (see 3.6.12). An outsider taps a communications line
An outsider uses a device to physically tap the communications line.
This attack is to be countered with support from the Environment.

7.1.21 Hacker masquerading as a legitimate user or as system process

T.Hack_Masq (see 3.5.6).
The following Detailed Attacks form this General Threat category:

DA.Hack_Masq_Hijack (see 3.4.34). A hacker assumes the identity of an authorized user
A hacker captures the interactive session of an authorized user. The hacker now appears as a legitimate user and can perform any action allowed to that user, including reading or modifying sensitive data.
This attack is countered solely by the TOE.

DA.Hack_Masq_Uwkstn (see 3.4.35). A user assumes the identity of an authorized user
An individual takes advantage of an unattended but active workstation to perform operations in the name of the logged-in user. Such operations may include some operations that the attacker is not normally allowed to perform.
This attack is countered solely by the TOE.

DA.Hack_Masq_Wauth (see 3.6.13). Masquerading due to weak authentication
Services are provided to a user application without adequate authentication of the client requesting the service. This would permit someone to receive services for which they are not authorized. However, the server would see them as a legitimate user, which is why this is classified as a masquerade attack.
This attack is to be countered with support from the Environment.

7.1.22 Exploitation of vulnerabilities in the physical environment of the system

T.Hack_Phys (see 3.5.7).
The following Detailed Attacks form this General Threat category:

DA.Hack_Phys_Crypto (see 3.6.14). Physical attack on cryptographic assets
Physical attack causes damage to cryptographic functions and/or release of cryptographic assets
This attack is to be countered with support from the Environment.

DA.Hack_Phys_Damage (see 3.6.15). Hacker physically attacks the system
Hacker physically attacks the system, causing physical damage and loss of security protection.
This attack is to be countered with support from the Environment.

7.1.23 Social engineering

T.Hack_Social_Engineer (see 3.5.8).
The following Detailed Attacks form this General Threat category:

DA.Hack_SocEng_Password (see 3.6.16). Social engineering to steal password

A hacker persuades a user or administrator to reveal his password, giving the hacker access to the person's account privileges.

This attack is to be countered with support from the Environment.

DA.Hack_SocEng_SysInfo (see 3.6.17). Hacker uses social engineering to learn system information

A hacker persuades a user or administrator to reveal information about system operational procedures, auditing and known flaws.

This attack is to be countered with support from the Environment.

7.1.24 A critical system component fails

T.Component_Failure (see 3.5.9).
The following Detailed Attacks form this General Threat category:

DA.Hardware_Flaw (see 3.6.18). System hardware fails during system operation

System use uncovers a hardware flaw in a critical system component.

This attack is to be countered with support from the Environment.

DA.Phys_CompFail_Res (see 3.4.42). Resource depletion failure

A system allocates so many resources that not enough are left for a critical component to function correctly.

This attack is countered solely by the TOE.

DA.Software_Flaw (see 3.6.19). System use uncovers an intrinsic software flaw in a critical system component

An authorized user performs an operation or set of operations, exercising a software flaw in a security-critical component.

This attack is to be countered with support from the Environment.

DA.TSF_Err_Conf_Crypto (see 3.4.47). Accidental release of cryptographic assets due to TSF flaw or malfunction

The TSF accidentally releases sensitive plaintext data, red keys, or other cryptographic assets to an inappropriate audience.

This attack is countered solely by the TOE.

7.1.25 Failure of a distributed system component

T.Failure_DS_Comp (see 3.5.10).
The following Detailed Attacks form this General Threat category:

DA.Failure_DS_Comm (see 3.6.20). Communications function failure

Failure of a communications function severs communications between security-critical (TSF) components.

This attack is to be countered with support from the Environment.

7.1.26 Recipient denies receiving information

T.Repudiate_Receive (see 3.5.11).
The following Detailed Attacks form this General Threat category:

DA.Repudiate_Rcvr_Int (see 3.6.21). Denial of having received data from another local user

A local, authorized user receives a message from another local user via the system, and then denies having received it. This typically affects the sender of the message who is counting on responsibilities associated with receipt of the message.

This attack is to be countered with support from the Environment.

DA.Repudiate_Rcvr_Local (see 3.6.22). Denial of having received information from a remote user

A local, authorized user receives a message from another user at a remote trusted product, and then denies having received it.

This attack is to be countered with support from the Environment.

DA.Repudiate_Rcvr_Rem (see 3.6.23). Denial of having received information by a remote user

A local, authorized user sends a message to another user at a remote trusted product who then denies having received it.

This attack is to be countered with support from the Environment.

7.1.27 A participant denies performing a transaction

T.Repudiate_Transact (see 3.5.12).
The following Detailed Attacks form this General Threat category:

DA.Repudiate_Trans_Loc (see 3.6.24). Circumvent non-repudiation in a transaction involving a user and a local system

An authorized user participates in a transaction by responding to system/application prompts and then denies that the dialogue took place. The user and system/application are collocated.

This attack is to be countered with support from the Environment.

DA.Repudiate_Trans_Uloc (see 3.6.25). Circumvent non-repudiation in a transaction involving a local user and a remote system

An authorized user participates in a transaction by responding to remote system/application prompts and then denies that the dialogue took place.

This attack is to be countered with support from the Environment.

DA.Repudiate_Trans_Urem (see 3.6.26). Circumvent non-repudiation in a transaction involving a remote user and a local system

An authorized remote user participates in a transaction by responding to local system/application prompts and then denies that the dialogue took place.

This attack is to be countered with support from the Environment.

7.1.28 User error makes data inaccessible

T.User_Err_Inaccess (see 3.5.13).
The following Detailed Attacks form this General Threat category:

DA.User_Err_Delete (see 3.6.27). User error deletes data

An authorized user accidentally deletes user data.

This attack is to be countered with support from the Environment.

DA.User_Err_Mod_Attr (see 3.4.61). User error modifying attributes availability

An authorized user erroneously modifies the initial security attributes of user data, which makes the data inaccessible.

This attack is countered solely by the TOE.

DA.User_Err_Set_Attr (see 3.4.64). User error setting attributes availability

An authorized user erroneously sets the initial security attributes of user data, which makes the data inaccessible.

This attack is countered solely by the TOE.

7.1.29 User's misuse causes denial of service

T.User_Misuse_Avl_Resc (see 3.5.14).

The following Detailed Attacks form this General Threat category:

DA.User_Comm_Overload (see 3.4.55). User's unauthorized use causes overload of communication resources

An authorized user exceeds the authorized use of communication resources during the system (TOE) operation. This causes a denial or delay in service to legitimate operations within the TOE scope of control.

This attack is countered solely by the TOE.

DA.User_ErrAvl_AudExhst (see 3.4.56). Denial of service due to exhausted audit storage

An authorized user's actions generate so many audit records that audit storage space is exhausted and the system subsequently denies further service until audit storage becomes available.

This attack is countered solely by the TOE.

DA.User_Obst_Res_Use (see 3.6.28). User obstructs legitimate use of resources.

An authorized user obstructs the use resources by unauthorized modification of data file, communication channel, or object security attributes.

This attack is to be countered with support from the Environment.

DA.User_Prcsr_Overload (see 3.4.69). User's unauthorized actions over-task the system causing processor overload

The system (TOE) has been over-tasked and can not complete the assigned tasking at all or in an expected amount of time. The user invokes processing functions in association with unauthorized activity that leads to overburdening processing resources on the TOE.

This attack is countered solely by the TOE.

DA.User_Stg_Overload (see 3.4.72). User's unauthorized actions cause storage overload

An authorized user's unauthorized use of data storage causes a shortage of disk space for other users.

This attack is countered solely by the TOE.

7.2 Attacks and Security Objectives correspondence.

Once General Threats are decomposed into Detailed Attacks, security Objectives are allocated to counter each one of these relevant Detailed Attacks. This section presents such mapping, where for each attack, the applicable Security Objectives are shown.

7.2.1 Accidental mismanagement of cryptographic functions

DA.Adm_Err_Crypto (see 3.4.1).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Account (see 4.1.38). Auditing for user accountability

Provide information about past user behavior to an authorized user through system mechanisms. Specifically, during any specified time interval, the system is able to report to a user acting in an identified audit role selected auditable actions that a user has performed, and as a result, what auditable objects were affected and what auditable information was received by that user.

This objetive is fullfilled by the TOE.

O.Crypto_Key_Man (see 4.1.55). Cryptographic Key Management

Fully define cryptographic components, functions, and interfaces. Ensure appropriate protection for cryptographic keys throughout their lifecycle, covering generation, distribution, storage, use, and destruction.

This objetive is fullfilled by the TOE.

O.Crypto_Manage_Roles (see 4.1.56). Management of cryptographic roles

Provide one or more roles to manage cryptographic assets and attributes.

This objetive is fullfilled by the TOE.

O.I&A_User_Action (see 4.1.74). User-action identification and authentication

Associate each user-requested action with the user who requested the action.

This objetive is fullfilled by the TOE.

7.2.2 Destruction or modification of audit data

DA.Adm_Hstl_Audit_Dstr (see 3.4.2).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Admin_Role (see 4.1.39). Audit-administration role duties

Deter modification or destruction of audit data through the creation of an audit-administration role.

This objetive is fullfilled by the TOE.

O.Audit_Protect (see 4.1.44). Protect stored audit records

Protect audit records against unauthorized access, modification, or deletion to ensure accountability of user actions.

This objetive is fullfilled by the TOE.

O.I&A_User (see 4.1.73). Identify and authenticate each user

Uniquely identify and authenticate each user of the system.

This objetive is fullfilled by the TOE.

7.2.3 Administrator modifies or destroys user data or applications

DA.Adm_Hstl_Mod_DataAps (see 3.4.3).
The following Security Objective(s) are considered sufficient to counter this attack:

O.AC_Admin_Limit (see 4.1.1). Limitation of administrative access control

Design administrative functions in such a way that administrators do not automatically have access to user objects, except for necessary exceptions. For an accounts administrator, the necessary exceptions include allocation and de-allocation of storage. For an audit administrator, the necessary exceptions include observation of audited actions. In general, the exceptions tend to be role specific.

This objetive is fullfilled by the TOE.

O.Audit_Admin_Role (see 4.1.39). Audit-administration role duties

Deter modification or destruction of audit data through the creation of an audit-administration role.

This objetive is fullfilled by the TOE.

O.I&A_User (see 4.1.73). Identify and authenticate each user

Uniquely identify and authenticate each user of the system.

This objetive is fullfilled by the TOE.

O.Trusted_Path&Channel (see 4.1.136). Trusted path and channel

Provide a trusted path to security-critical (TSF) data in which both end points have assured identities. For the remote user, there needs to be a trusted channel as well.

This objetive is fullfilled by the TOE.

7.2.4 Administrator maliciously modifies or deletes data access control attributes

DA.Adm_Hstl_Mod_Data_AC (see 3.4.4).
The following Security Objective(s) are considered sufficient to counter this attack:

O.AC_Admin_Limit (see 4.1.1). Limitation of administrative access control

Design administrative functions in such a way that administrators do not automatically have access to user objects, except for necessary exceptions. For an accounts administrator, the necessary exceptions include allocation and de-allocation of storage. For an audit administrator, the necessary exceptions include observation of audited actions. In general, the exceptions tend to be role specific.

This objetive is fullfilled by the TOE.

O.Audit_Account (see 4.1.38). Auditing for user accountability

Provide information about past user behavior to an authorized user through system mechanisms. Specifically, during any specified time interval, the system is able to report to a user acting in an identified audit role selected auditable actions that a user has performed, and as a result, what auditable objects were affected and what auditable information was received by that user.

This objetive is fullfilled by the TOE.

O.Audit_Loss_Respond (see 4.1.43). Respond to possible loss of stored audit records

Respond to possible loss of audit records when audit trail storage is full or nearly full.

This objetive is fullfilled by the TOE.

O.Audit_Protect (see 4.1.44). Protect stored audit records

Protect audit records against unauthorized access, modification, or deletion to ensure accountability of user actions.

This objetive is fullfilled by the TOE.

7.2.5 The administrator maliciously modifies information flow control.

DA.Adm_Hstl_Mod_IFC (see 3.4.5).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Admin_Role (see 4.1.39). Audit-administration role duties

Deter modification or destruction of audit data through the creation of an audit-administration role.

This objetive is fullfilled by the TOE.

O.Audit_Protect (see 4.1.44). Protect stored audit records

Protect audit records against unauthorized access, modification, or deletion to ensure accountability of user actions.

This objetive is fullfilled by the TOE.

O.I&A_User (see 4.1.73). Identify and authenticate each user

Uniquely identify and authenticate each user of the system.

This objetive is fullfilled by the TOE.

O.Info_Flow_Ctrl_Admin (see 4.1.77). Provide information flow control administration

Manage information flow control policy and functions to allow only specified administrators to have the ability to manipulate the information flow control.

This objetive is fullfilled by the TOE.

7.2.6 Administrator maliciously modifies system entry parameters

DA.Adm_Hstl_Mod_SEP (see 3.4.6).
The following Security Objective(s) are considered sufficient to counter this attack:

O.AC_Admin_Limit (see 4.1.1). Limitation of administrative access control

Design administrative functions in such a way that administrators do not automatically have access to user objects, except for necessary exceptions. For an accounts administrator, the necessary exceptions include allocation and de-allocation of storage. For an audit administrator, the necessary exceptions include observation of audited actions. In general, the exceptions tend to be role specific.

This objetive is fullfilled by the TOE.

O.Aud_Sys_Entry_Parms (see 4.1.37). Audit changes of system entry parameters

Deter an administrator from changing system entry parameters to allow an unauthorized user access to organizational assets to which they are forbidden.

This objetive is fullfilled by the TOE.

O.Audit_Admin_Role (see 4.1.39). Audit-administration role duties

Deter modification or destruction of audit data through the creation of an audit-administration role.

This objetive is fullfilled by the TOE.

O.I&A_User (see 4.1.73). Identify and authenticate each user

Uniquely identify and authenticate each user of the system.

This objetive is fullfilled by the TOE.

7.2.7 Administrator maliciously modifies security-critical code

DA.Adm_Hstl_Mod_TSFCode (see 3.4.7).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Obj_Protection (see 4.1.103). Object domain protection

Require domain protection for objects. Specify object classes (domains), user groups, and operation classes. Use these to specify which operations may be performed on which objects by which users. Basically this controls what users can do in a given group.

This objetive is fullfilled by the TOE.

O.Sys_Self_Protection (see 4.1.128). Protection of system security function

Protect the system security functions through technical features.

This objetive is fullfilled by the TOE.

O.TSF_Mod_Limit (see 4.1.129). Limit administrator's modification of security-critical code or data

Limit the malicious modification of security-critical (TSF) code and data to include specific system code to prevent the system security protection capabilities from being diminished or weakened.

This objetive is fullfilled by the TOE.

7.2.8 Administrator maliciously modifies user/subject bindings

DA.Adm_Hstl_Mod_USB (see 3.4.8).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Adm_Limits_Bindings (see 4.1.4). Limit an administrator's ability to modify user-subject bindings

Limit the administrator from modification of user-subject bindings in an effort to deter users acting without accountability.

This objetive is fullfilled by the TOE.

7.2.9 Administrator maliciously modifies user attributes and/or roles

DA.Adm_Hstl_Mod_UsrAttr (see 3.4.9).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Adm_User_Att_Mod (see 4.1.5). Limit administrator's modification of user attributes

Deter the administrator from maliciously modifying users' attributes. Such modifications could allow unauthorized user actions or denial of service to a legitimate user.

This objetive is fullfilled by the TOE.

7.2.10 User privileges and/or authorizations are not updated upon reassignment

DA.Adm_Misconfig_User (see 3.4.10).
The following Security Objective(s) are considered sufficient to counter this attack:

O.User_Auth_Management (see 4.1.140). User authorization management

Manage and update user authorization and privilege data in accordance with organizational security and personnel policies.

This objetive is fullfilled by the TOE.

7.2.11 Administrator error modifies access control or information flow policy

DA.Admin_Err_AC_Policy (see 3.4.11).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Guidance (see 4.1.8). Administrator guidance documentation

Deter administrator errors by providing adequate administrator guidance.

This objetive is fullfilled by the TOE.

O.Security_Attr_Mgt (see 4.1.114). Manage security attributes

Manage the initialization of, values for, and allowable operations on security attributes.

This objetive is fullfilled by the TOE.

O.Security_Data_Mgt (see 4.1.115). Manage security-critical data

Manage the initialization of, limits on, and allowable operations on security-critical data.

This objetive is fullfilled by the TOE.

O.Security_Func_Mgt (see 4.1.116). Manage behavior of security functions

Provide management mechanisms for security mechanisms.

This objetive is fullfilled by the TOE.

O.Security_Roles (see 4.1.117). Security roles

Maintain security-relevant roles and the association of users with those roles.

This objetive is fullfilled by the TOE.

7.2.12 Administrator error changes audit behavior

DA.Admin_Err_Audit (see 3.4.12).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Guidance (see 4.1.8). Administrator guidance documentation

Deter administrator errors by providing adequate administrator guidance.

This objetive is fullfilled by the TOE.

O.Audit_Admin_Role (see 4.1.39). Audit-administration role duties

Deter modification or destruction of audit data through the creation of an audit-administration role.

This objetive is fullfilled by the TOE.

O.Audit_Loss_Respond (see 4.1.43). Respond to possible loss of stored audit records

Respond to possible loss of audit records when audit trail storage is full or nearly full.

This objetive is fullfilled by the TOE.

O.Audit_Protect (see 4.1.44). Protect stored audit records

Protect audit records against unauthorized access, modification, or deletion to ensure accountability of user actions.

This objetive is fullfilled by the TOE.

O.I&A_User (see 4.1.73). Identify and authenticate each user

Uniquely identify and authenticate each user of the system.

This objetive is fullfilled by the TOE.

O.Security_Data_Mgt (see 4.1.115). Manage security-critical data

Manage the initialization of, limits on, and allowable operations on security-critical data.

This objetive is fullfilled by the TOE.

O.Security_Roles (see 4.1.117). Security roles

Maintain security-relevant roles and the association of users with those roles.

This objetive is fullfilled by the TOE.

7.2.13 Administrator error modifies authentication enforcement

DA.Admin_Err_Authentic (see 3.4.13).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Guidance (see 4.1.8). Administrator guidance documentation

Deter administrator errors by providing adequate administrator guidance.

This objetive is fullfilled by the TOE.

O.Limit_Actions_Auth (see 4.1.89). Restrict actions before authentication

Restrict the actions a user may perform before the TOE verifies the identity of the user.

This objetive is fullfilled by the TOE.

O.Security_Data_Mgt (see 4.1.115). Manage security-critical data

Manage the initialization of, limits on, and allowable operations on security-critical data.

This objetive is fullfilled by the TOE.

7.2.14 Administrator error makes information unavailable

DA.Admin_Err_Info (see 3.4.14).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Guidance (see 4.1.8). Administrator guidance documentation
Deter administrator errors by providing adequate administrator guidance.
This objetive is fullfilled by the TOE.

O.Priority_Of_Service (see 4.1.104). Provide priority of service
Control access to resources so that lower-priority activities do not unduly interfere with or delay higher-priority activities.
This objetive is fullfilled by the TOE.

O.Resource_Quotas (see 4.1.110). Resource quotas for users and services
Use resource quotas to limit user and service use of system resources to a level that will prevent degradation or denial of service to other critical users and services.
This objetive is fullfilled by the TOE.

O.Security_Data_Mgt (see 4.1.115). Manage security-critical data
Manage the initialization of, limits on, and allowable operations on security-critical data.
This objetive is fullfilled by the TOE.

O.Security_Roles (see 4.1.117). Security roles
Maintain security-relevant roles and the association of users with those roles.
This objetive is fullfilled by the TOE.

7.2.15 Back door left open

DA.Admin_Err_Omit_Trap (see 3.4.15).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Maintenance_Access (see 4.1.93). Controlled access by maintenance personnel
Control access to the system by maintenance personnel who troubleshoot the system and perform system updates.
This objetive is fullfilled by the TOE.

O.Maintenance_Recover (see 4.1.94). Expiration of maintenance privileges
Terminate maintenance user system access privilege automatically after expiration of assigned timed interval.
This objetive is fullfilled by the TOE.

O.Prvlg_IF_Status (see 4.1.105). Privileged-interface status
Provide capability for an administrator to determine the use status of all privileged interfaces. This would include interfaces used by maintenance personnel.
This objetive is fullfilled by the TOE.

7.2.16 Administrator error makes resource unavailable

DA.Admin_Err_Resource (see 3.4.16).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Guidance (see 4.1.8). Administrator guidance documentation
Deter administrator errors by providing adequate administrator guidance.
This objetive is fullfilled by the TOE.

O.Security_Attr_Mgt (see 4.1.114). Manage security attributes
Manage the initialization of, values for, and allowable operations on security attributes.
This objetive is fullfilled by the TOE.

7.2.17 Administrator error modifies entry policy

DA.Admin_Err_Sys_Entry (see 3.4.17).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Guidance (see 4.1.8). Administrator guidance documentation
Deter administrator errors by providing adequate administrator guidance.
This objetive is fullfilled by the TOE.

O.Security_Data_Mgt (see 4.1.115). Manage security-critical data
Manage the initialization of, limits on, and allowable operations on security-critical data.
This objetive is fullfilled by the TOE.

7.2.18 Administrator fails to update security configuration

DA.Admin_Err_Update (see 3.4.18).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Guidance (see 4.1.8). Administrator guidance documentation
Deter administrator errors by providing adequate administrator guidance.
This objetive is fullfilled by the TOE.

O.Audit_Account (see 4.1.38). Auditing for user accountability
Provide information about past user behavior to an authorized user through system mechanisms. Specifically, during any specified time interval, the system is able to report to a user acting in an identified audit role selected auditable actions that a user has performed, and as a result, what auditable objects were affected and what auditable information was received by that user.
This objetive is fullfilled by the TOE.

O.Secure_Configuration (see 4.1.112). Security-relevant configuration management
Manage and update system security policy data and enforcement functions, and other security-relevant configuration data, in accordance with organizational security policies.
This objetive is fullfilled by the TOE.

7.2.19 Administrator error modifies user security attributes

DA.Admin_Err_User_Attr (see 3.4.19).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Guidance (see 4.1.8). Administrator guidance documentation
Deter administrator errors by providing adequate administrator guidance.
This objetive is fullfilled by the TOE.

O.Security_Attr_Mgt (see 4.1.114). Manage security attributes
Manage the initialization of, values for, and allowable operations on security attributes.
This objetive is fullfilled by the TOE.

O.Security_Data_Mgt (see 4.1.115). Manage security-critical data
Manage the initialization of, limits on, and allowable operations on security-critical data.
This objetive is fullfilled by the TOE.

O.Security_Func_Mgt (see 4.1.116). Manage behavior of security functions
Provide management mechanisms for security mechanisms.
This objetive is fullfilled by the TOE.

O.Security_Roles (see 4.1.117). Security roles
Maintain security-relevant roles and the association of users with those roles.
This objetive is fullfilled by the TOE.

O.User_Attributes (see 4.1.139). Maintain user attributes

Maintain a set of security attributes (which may include group membership, clearance, access rights, etc.) associated with individual users in addition to user identity.

This objetive is fullfilled by the TOE.

7.2.20 Administrator aggregates privacy information

DA.Admin_UserPriv_Agg (see 3.6.1).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Limit_ObserveRoles (see 4.2.5). Limit observation of service usage to authorized users

Provide authorized users with the capability to observe the usage of specified services or resources as necessary to perform their duties.

This objetive is assumed to be fulfilled by the environment.

O.Prevent_Link (see 4.2.14). Prevent linking of multiple service use

Ensure that a user may make multiple uses of a service or resource without other specified users being able to link these uses together.

This objetive is assumed to be fulfilled by the environment.

O.Prevent_Observe (see 4.2.15). Prevent observation of service use

Ensure that a user may use a service or resource without other specified users being able to observe that the service or resource is being used.

This objetive is assumed to be fulfilled by the environment.

7.2.21 Administrator reads collected user privacy information

DA.Admin_UserPriv_Col (see 3.6.2).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Prevent_AskPrivInfo (see 4.2.13). Prevent system from collecting user privacy information

Provide some services or resources to specified users without soliciting from the user information that is relevant to the user's privacy.

This objetive is assumed to be fulfilled by the environment.

7.2.22 Administrator reads system generated privacy information

DA.Admin_UserPriv_Gen (see 3.6.3).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Permit_Aliases (see 4.2.11). Permit users to use services under aliases

Permit some users to maintain partial anonymity when using specified services or resources by means of aliases.

This objetive is assumed to be fulfilled by the environment.

O.Permit_Anonymity (see 4.2.12). Permit users to use services anonymously

Permit some users to maintain anonymity when using specified services or resources.

This objetive is assumed to be fulfilled by the environment.

7.2.23 Inconsistent interpretation of audit data attributes

DA.Dev_FC_Attr_Interp (see 3.4.20).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Integrity_Attr_Exch (see 4.1.83). Correct attribute exchange with another trusted product

Ensure that the system correctly exchanges security-attribute information with another trusted IT product.

This objetive is fullfilled by the TOE.

7.2.24 Buffers not cleared by the system

DA.Dev_FC_Buff_Not_Clr (see 3.4.21).
The following Security Objective(s) are considered sufficient to counter this attack:

O.No_Residual_Info (see 4.1.97). Eliminate residual information

Ensure there is no object reuse; i.e., ensure that there is no residual information in some information containers or system resources upon their reallocation to different users.

This objetive is fullfilled by the TOE.

7.2.25 Incorrect modification of control data

DA.Dev_FC_Ctrl_Data (see 3.4.22).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Integ_Sys_Data_Int (see 4.1.81). Integrity of system data transferred internally

Ensure the integrity of system data transferred internally.

This objetive is fullfilled by the TOE.

7.2.26 System data incorrectly exchanged

DA.Dev_FC_Data_Export (see 3.4.23).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Integ_Sys_Data_Ext (see 4.1.80). Integrity of system data transferred externally

Ensure the integrity of system data exchanged externally with another trusted product by using a protocol for data transfer that will permit error detection and correction. This includes detecting and possibly correcting errors in data received and encoding outgoing data to make it possible for the receiver to detect and possibly correct errors. The method for detecting and correcting errors is based on some method (protocol) that is agreed upon by participating parties.

This objetive is fullfilled by the TOE.

7.2.27 Non-secure recovery

DA.Dev_FC_Recovery (see 3.4.24).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Secure_State (see 4.1.113). Protect and maintain secure system state

Maintain and recover to a secure state without security compromise after system error or other interruption of system operation.

This objetive is fullfilled by the TOE.

7.2.28 Inaccurate system-data replication

DA.Dev_FC_Replication (see 3.4.25).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Integrity_Data_Rep (see 4.1.85). Integrity of system data replication

Ensure that when system data replication occurs across the system the data is consistent for each replication.

This objetive is fullfilled by the TOE.

7.2.29 System modification by unauthorized source

DA.Dev_FC_Self_Protect (see 3.4.26).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Correct_Operation (see 4.1.49). Verify correct operation as designed

Provide the ability for the authorized user to verify that the system operates as designed.

This objetive is fullfilled by the TOE.

O.Sys_Self_Protection (see 4.1.128). Protection of system security function

Protect the system security functions through technical features.

This objetive is fullfilled by the TOE.

7.2.30 Malicious developer creates secret trapdoor in system

DA.Dev_FC_Trap_Door (see 3.4.27).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Account (see 4.1.38). Auditing for user accountability

Provide information about past user behavior to an authorized user through system mechanisms. Specifically, during any specified time interval, the system is able to report to a user acting in an identified audit role selected auditable actions that a user has performed, and as a result, what auditable objects were affected and what auditable information was received by that user.

This objetive is fullfilled by the TOE.

O.Audit_Admin_Role (see 4.1.39). Audit-administration role duties

Deter modification or destruction of audit data through the creation of an audit-administration role.

This objetive is fullfilled by the TOE.

O.Code_Signing (see 4.1.46). Code signing and verification

Check verification of signed downloaded code prior to execution. A well-known example is checking digital signatures on signed Java applets.

This objetive is fullfilled by the TOE.

O.I&A_User (see 4.1.73). Identify and authenticate each user

Uniquely identify and authenticate each user of the system.

This objetive is fullfilled by the TOE.

O.Source_Code_Exam (see 4.1.121). Examine the source code for developer flaws

Examine for accidental or deliberate flaws in code made by the developer. The accidental flaws could be lack of engineering detail or bad design. Where the deliberate flaws would include building trapdoors for later entry as an example.

This objetive is fullfilled by the TOE.

7.2.31 Communications function failure

DA.Failure_DS_Comm (see 3.6.20).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Fault_Tolerance (see 4.2.4). Provide fault tolerant operations for critical components

Provide fault tolerant operations for critical components and continue to operate in the presence of specific failures in one or more system components.

This objetive is assumed to be fulfilled by the environment.

O.Integrity_Data_Rep (see 4.1.85). Integrity of system data replication

Ensure that when system data replication occurs across the system the data is consistent for each replication.

This objetive is fullfilled by the TOE.

O.Trusted_DS_Recov (see 4.1.134). Trusted distributed system recovery

Ensure that a replaced failed component when re-integrated into the system will recover such that it will not cause errors or security breaches in other parts of the system.

This objetive is fullfilled by the TOE.

7.2.32 Hacker gains access through a vulnerability in code

DA.Hack_AC_Code_Vul (see 3.4.28).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Apply_Code_Fixes (see 4.1.9). Apply patches to fix the code

Apply patches to fix the code when vulnerabilities in code allow unauthorized and undiscovered access.

This objetive is fullfilled by the TOE.

O.Audit_Deter_Misuse (see 4.1.40). Audit system access to deter misuse

Audit system access to discover system misuse and provide a potential deterrent by warning the user.

This objetive is fullfilled by the TOE.

7.2.33 Weak system access control mechanism or system access control implementation

DA.Hack_AC_Weak (see 3.4.29).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Hack_Limit_Sessions (see 4.1.69). Limit sessions to outside users

Limit the number of sessions available to outside users. A hacker can initiate multiple communication sessions that could cause an overload on resources, for example, half open session starts as is seen in SYN flood attacks.

This objetive is fullfilled by the TOE.

O.Trusted_Path (see 4.1.135). Provide a trusted path

Provide a trusted path between the user and the system. Execution of a user-requested action must be made via a trusted path with the following properties:

- The path is logically distinct from, and cannot be confused with other communication paths (by either the user or the system).
- The path provides assured identification of its end points.

This objetive is fullfilled by the TOE.

7.2.34 The communication mechanism emanates data

DA.Hack_CommEaves_Eman (see 3.4.30).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Data_Exchange_Conf (see 4.1.61). Enforce data exchange confidentiality

Protect user data confidentiality when exchanging data with a remote system.

This objetive is fullfilled by the TOE.

7.2.35 Outsider intercepts user communications

DA.Hack_CommEaves_Intrc (see 3.4.31).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Data_Exchange_Conf (see 4.1.61). Enforce data exchange confidentiality
Protect user data confidentiality when exchanging data with a remote system.
This objetive is fullfilled by the TOE.

7.2.36 An outsider taps a communications line

DA.Hack_CommEaves_Tap (see 3.6.12).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Comm_Line_Protection (see 4.2.3). Physical protection of the communications line
Protect communications lines from physical tampering.
This objetive is assumed to be fulfilled by the environment.

O.Tamper_ID (see 4.2.19). Tamper detection
Provide system features that detect physical tampering of a system component, and use those features to limit security breaches.
This objetive is assumed to be fulfilled by the environment.

O.Tamper_Resistance (see 4.2.20). Tamper resistance
Prevent or resist physical tampering with specified system devices and components.
This objetive is assumed to be fulfilled by the environment.

7.2.37 Hacker causes overload of communication resources

DA.Hack_Comm_Overload (see 3.4.32).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Generation (see 4.1.42). Audit records with identity
Record in audit records: date and time of action, location of the action, and the entity responsible for the action.
This objetive is fullfilled by the TOE.

O.Data_Imp_Exp_Control (see 4.1.63). Data import/export to/from system control
Protect data from being sent to erroneous places and more places external to the system than allowed by the organization's security policy. Conversely the import of data into the system should be protected from illicit information or information not allowed by the organization's security policy.
This objetive is fullfilled by the TOE.

O.Hack_Limit_Sessions (see 4.1.69). Limit sessions to outside users
Limit the number of sessions available to outside users. A hacker can initiate multiple communication sessions that could cause an overload on resources, for example, half open session starts as is seen in SYN flood attacks.
This objetive is fullfilled by the TOE.

O.Hack_Traffic_Control (see 4.1.70). Control hacker communication traffic
Control (e.g. reroute or discard) hacker communication traffic to prevent potential damage.
This objetive is fullfilled by the TOE.

O.Priority_Of_Service (see 4.1.104). Provide priority of service
Control access to resources so that lower-priority activities do not unduly interfere with or delay higher-priority activities.
This objetive is fullfilled by the TOE.

O.Resource_Quotas (see 4.1.110). Resource quotas for users and services

Use resource quotas to limit user and service use of system resources to a level that will prevent degradation or denial of service to other critical users and services.

This objetive is fullfilled by the TOE.

7.2.38 Accidental or deliberate mishandling of cryptographic assets external to the TOE

DA.Hack_Ext_CryptoAsset (see 3.4.33).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Crypto_Import_Export (see 4.1.54). Cryptographic import, export, and inter-TSF transfer

Protect cryptographic data assets when they are being transmitted to and from the TOE, either through intervening untrusted components or directly to/from human users.

This objetive is fullfilled by the TOE.

O.Crypto_Manage_Roles (see 4.1.56). Management of cryptographic roles

Provide one or more roles to manage cryptographic assets and attributes.

This objetive is fullfilled by the TOE.

7.2.39 A hacker assumes the identity of an authorized user

DA.Hack_Masq_Hijack (see 3.4.34).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Gen_User (see 4.1.41). Individual accountability

Provide individual accountability for audited events. Uniquely identify each user so that auditable actions can be traced to a user.

This objetive is fullfilled by the TOE.

O.Trusted_Path (see 4.1.135). Provide a trusted path

Provide a trusted path between the user and the system. Execution of a user-requested action must be made via a trusted path with the following properties:

- The path is logically distinct from, and cannot be confused with other communication paths (by either the user or the system).
- The path provides assured identification of its end points.

This objetive is fullfilled by the TOE.

7.2.40 A user assumes the identity of an authorized user

DA.Hack_Masq_Uwkstn (see 3.4.35).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Gen_User (see 4.1.41). Individual accountability

Provide individual accountability for audited events. Uniquely identify each user so that auditable actions can be traced to a user.

This objetive is fullfilled by the TOE.

O.Screen_Lock (see 4.1.111). User screen locking

Provide a screen lock function to prevent an unauthorized user from using an unattended computer where a valid user has an active session.

This objetive is fullfilled by the TOE.

O.Session_Termination (see 4.1.118). System terminates session for inactivity

System terminates a session after a given interval of inactivity.

This objetive is fullfilled by the TOE.

O.Trusted_Path (see 4.1.135). Provide a trusted path

Provide a trusted path between the user and the system. Execution of a user-requested action must be made via a trusted path with the following properties:

- The path is logically distinct from, and cannot be confused with other communication paths (by either the user or the system).
- The path provides assured identification of its end points.

This objetive is fullfilled by the TOE.

O.User_Guidance (see 4.1.146). User guidance documentation

Provide documentation for the general user.

This objetive is fullfilled by the TOE.

7.2.41 Masquerading due to weak authentication

DA.Hack_Masq_Wauth (see 3.6.13).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Generation (see 4.1.42). Audit records with identity

Record in audit records: date and time of action, location of the action, and the entity responsible for the action.

This objetive is fullfilled by the TOE.

O.User_Auth_Enhanced (see 4.2.21). Enhanced user authentication

Execute enhanced measures to ensure that either user authentication data cannot be stolen or when it is stolen, it cannot be used to gain access to the system.

This objetive is assumed to be fulfilled by the environment.

O.User_Auth_Multiple (see 4.2.22). Require multiple authentication mechanisms

Invoke multiple authentication mechanisms, which will provide confidence that the user is who they say they are.

This objetive is assumed to be fulfilled by the environment.

7.2.42 Modification of security-critical data in transit from a remote trusted site

DA.Hack_MsgData_RcvTSF (see 3.4.36).
The following Security Objective(s) are considered sufficient to counter this attack:

O.TSF_Rcv_Err_ID_Loc (see 4.1.130). Local detection of received security-critical data modified in transit

Identification by the system (TOE) of modification of security-critical (TSF) data occurring in transit from a remote trusted site must occur.

This objetive is fullfilled by the TOE.

O.TSF_Rcv_Err_ID_Rem (see 4.1.131). Remote detection of received security-critical data modified in transit

Identification by the remote site of the modification of security-critical (TSF) data occurring in transit from the remote site must occur.

This objetive is fullfilled by the TOE.

7.2.43 Modification of user data in transit from a remote site

DA.Hack_MsgData_RcvUsr (see 3.4.37).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Rcv_MsgMod_ID (see 4.1.106). Identify message modification in messages received

The TSF recognizes changes to messages that occurred in transit, including insertion of spurious messages and deletion or replay of legitimate messages.

This objetive is fullfilled by the TOE.

O.Rcv_MsgMod_Rcvr (see 4.1.107). Recovery from modification of received messages

The TSF detects and corrects changes in messages received from a remote trusted site.

This objetive is fullfilled by the TOE.

7.2.44 Modification of security-critical data in transit to a remote site

DA.Hack_MsgData_SndTSF (see 3.4.38).

The following Security Objective(s) are considered sufficient to counter this attack:

O.TSF_Snd_Err_ID_Loc (see 4.1.132). Local detection of sent security-critical data modified in transit

Identification of modification of security-critical (TSF) data occurring in transit to a remote site by the TSF must occur.

This objetive is fullfilled by the TOE.

O.TSF_Snd_Err_ID_Rem (see 4.1.133). Remote detection of sent security-critical data modified in transit.

Identification of modification of security-critical (TSF) data occurring in transit to a remote site by the remote site must occur.

This objetive is fullfilled by the TOE.

7.2.45 Modification of user data in transit to a remote site

DA.Hack_MsgData_SndUsr (see 3.4.39).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Snt_MsgMod_ID (see 4.1.119). Identify message modification in messages sent

The TSF supports recognition of changes to transmitted messages that occurred in transit, including insertion of spurious messages and deletion or replay of legitimate messages.

This objetive is fullfilled by the TOE.

O.Snt_MsgMod_Rcvr (see 4.1.120). Support recovery from modification of sent messages

The TSF supports detection of changes in messages sent to a remote trusted site.

This objetive is fullfilled by the TOE.

7.2.46 Physical attack on cryptographic assets

DA.Hack_Phys_Crypto (see 3.6.14).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Tamper_ID (see 4.2.19). Tamper detection

Provide system features that detect physical tampering of a system component, and use those features to limit security breaches.

This objetive is assumed to be fulfilled by the environment.

O.Tamper_Resistance (see 4.2.20). Tamper resistance

Prevent or resist physical tampering with specified system devices and components.

This objetive is assumed to be fulfilled by the environment.

7.2.47 Hacker physically attacks the system

DA.Hack_Phys_Damage (see 3.6.15).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Tamper_ID (see 4.2.19). Tamper detection

Provide system features that detect physical tampering of a system component, and use those features to limit security breaches.

This objetive is assumed to be fulfilled by the environment.

O.Tamper_Resistance (see 4.2.20). Tamper resistance

Prevent or resist physical tampering with specified system devices and components.

This objetive is assumed to be fulfilled by the environment.

7.2.48 Hacker causes system task overload resulting in denial of service

DA.Hack_Prcsr_Overload (see 3.6.11).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Generation (see 4.1.42). Audit records with identity

Record in audit records: date and time of action, location of the action, and the entity responsible for the action.

This objetive is fullfilled by the TOE.

O.Hack_Limit_Sessions (see 4.1.69). Limit sessions to outside users

Limit the number of sessions available to outside users. A hacker can initiate multiple communication sessions that could cause an overload on resources, for example, half open session starts as is seen in SYN flood attacks.

This objetive is fullfilled by the TOE.

O.Hack_Traffic_Control (see 4.1.70). Control hacker communication traffic

Control (e.g. reroute or discard) hacker communication traffic to prevent potential damage.

This objetive is fullfilled by the TOE.

O.Priority_Of_Service (see 4.1.104). Provide priority of service

Control access to resources so that lower-priority activities do not unduly interfere with or delay higher-priority activities.

This objetive is fullfilled by the TOE.

O.React_Discovered_Atk (see 4.2.16). React to discovered attacks

Implement automated notification or other reactions to the TSF-discovered attacks in an effort to identify attacks and to create an attack deterrent.

This objetive is assumed to be fulfilled by the environment.

O.Resource_Quotas (see 4.1.110). Resource quotas for users and services

Use resource quotas to limit user and service use of system resources to a level that will prevent degradation or denial of service to other critical users and services.

This objetive is fullfilled by the TOE.

7.2.49 Social engineering to steal password

DA.Hack_SocEng_Password (see 3.6.16).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Limit_Mult_Sessions (see 4.1.91). Limit multiple sessions

Provide the capability to limit the number of sessions that a user may have open at one time.

This objetive is fullfilled by the TOE.

O.User_Auth_Enhanced (see 4.2.21). Enhanced user authentication

Execute enhanced measures to ensure that either user authentication data cannot be stolen or when it is stolen, it cannot be used to gain access to the system.

This objetive is assumed to be fulfilled by the environment.

7.2.50 Hacker uses social engineering to learn system information

DA.Hack_SocEng_SysInfo (see 3.6.17).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Guidance (see 4.1.8). Administrator guidance documentation

Deter administrator errors by providing adequate administrator guidance.

This objetive is fullfilled by the TOE.

O.Audit_Unusual_User (see 4.2.1). Audit unusual user activity

Audit unusual user activity.

This objetive is assumed to be fulfilled by the environment.

O.Identify_Unusual_Act (see 4.1.75). Identify unusual user activity

Identify unusual user activity on the system.

This objetive is fullfilled by the TOE.

O.User_Guidance (see 4.1.146). User guidance documentation

Provide documentation for the general user.

This objetive is fullfilled by the TOE.

7.2.51 Login program replicated to capture authentication data

DA.Hack_Spoof_Login (see 3.6.10).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Trusted_Path (see 4.1.135). Provide a trusted path

Provide a trusted path between the user and the system. Execution of a user-requested action must be made via a trusted path with the following properties:

- The path is logically distinct from, and cannot be confused with other communication paths (by either the user or the system).
- The path provides assured identification of its end points.

This objetive is fullfilled by the TOE.

O.User_Auth_Enhanced (see 4.2.21). Enhanced user authentication

Execute enhanced measures to ensure that either user authentication data cannot be stolen or when it is stolen, it cannot be used to gain access to the system.

This objetive is assumed to be fulfilled by the environment.

7.2.52 Attacker modifies protocol headers

DA.Hack_Spoof_MsgHdr (see 3.4.40).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Comm_Trusted_Channel (see 4.1.47). Trusted channel to remote trusted system

Provide a communications channel between the system and a remote trusted system for the performance of security-critical operations.

This objetive is fullfilled by the TOE.

O.Crypto_Comm_Channel (see 4.1.51). Encrypted communications channel

Provide secure session establishment between the system and remote systems using encryption functions.

This objetive is fullfilled by the TOE.

O.Integrity_Attr_Exch (see 4.1.83). Correct attribute exchange with another trusted product

Ensure that the system correctly exchanges security-attribute information with another trusted IT product.

This objetive is fullfilled by the TOE.

O.NonRepudiate_Sent (see 4.1.101). Non-repudiation for sent information

Provide evidence that a user sent information.

This objetive is fullfilled by the TOE.

7.2.53 Hacker activities cause storage overload

DA.Hack_Stg_Overload (see 3.4.41).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Generation (see 4.1.42). Audit records with identity

Record in audit records: date and time of action, location of the action, and the entity responsible for the action.

This objetive is fullfilled by the TOE.

O.Guarantee_Audit_Stg (see 4.1.68). Guarantee the availability of audit storage space

Maintain audit data and guarantee space for that data.

This objetive is fullfilled by the TOE.

O.Hack_Limit_Sessions (see 4.1.69). Limit sessions to outside users

Limit the number of sessions available to outside users. A hacker can initiate multiple communication sessions that could cause an overload on resources, for example, half open session starts as is seen in SYN flood attacks.

This objetive is fullfilled by the TOE.

O.Hack_Traffic_Control (see 4.1.70). Control hacker communication traffic

Control (e.g. reroute or discard) hacker communication traffic to prevent potential damage.

This objetive is fullfilled by the TOE.

O.Manage_TSF_Data (see 4.1.96). Manage security-critical data to avoid storage space being exceeded

Manage security-critical (TSF) data to ensure that the size of the data does not exceed the space allocated for storage of the data.

This objetive is fullfilled by the TOE.

O.Priority_Of_Service (see 4.1.104). Provide priority of service

Control access to resources so that lower-priority activities do not unduly interfere with or delay higher-priority activities.

This objetive is fullfilled by the TOE.

O.Resource_Quotas (see 4.1.110). Resource quotas for users and services

Use resource quotas to limit user and service use of system resources to a level that will prevent degradation or denial of service to other critical users and services.

This objetive is fullfilled by the TOE.

7.2.54 System hardware fails during system operation

DA.Hardware_Flaw (see 3.6.18).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Fail_Secure (see 4.1.66). Preservation of secure state for failures in critical components

Preserve the secure state of the system in the event of a secure component failure.

This objetive is fullfilled by the TOE.

O.Fault_Tolerance (see 4.2.4). Provide fault tolerant operations for critical components

Provide fault tolerant operations for critical components and continue to operate in the presence of specific failures in one or more system components.

This objetive is assumed to be fulfilled by the environment.

7.2.55 Malicious code perpetrator dissemination

DA.Mal_Code_Hack_Downld (see 3.6.4).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Clean_Obj_Recovery (see 4.2.2). Object and data recovery free from malicious code

Recover to a viable state after malicious code is introduced and damage occurs, removing the malicious code as part of the process.

This objetive is assumed to be fulfilled by the environment.

O.Code_Signing (see 4.1.46). Code signing and verification

Check verification of signed downloaded code prior to execution. A well-known example is checking digital signatures on signed Java applets.

This objetive is fullfilled by the TOE.

O.General_Integ_Checks (see 4.1.67). Periodically check integrity

Provide periodic integrity checks on both system and user data.

This objetive is fullfilled by the TOE.

O.Input_Inspection (see 4.1.78). Require inspection for absence of malicious code.

Require inspection of downloads/transfers.

This objetive is fullfilled by the TOE.

O.Obj_Protection (see 4.1.103). Object domain protection

Require domain protection for objects. Specify object classes (domains), user groups, and operation classes. Use these to specify which operations may be performed on which objects by which users. Basically this controls what users can do in a given group.

This objetive is fullfilled by the TOE.

O.Remote_Execution (see 4.1.109). Disable remote execution

Disable a remote entity's ability to execute local code.

This objetive is fullfilled by the TOE.

7.2.56 Malicious code perpetrator execution

DA.Mal_Code_Hack_Exe (see 3.6.5).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Code_Val (see 4.1.6). Administrative validation of executables

Validate executable objects prior to allowing execution. Validation needs to be done by someone with an expertise to recognize malicious code and the authority and means to prevent its execution.

This objetive is fullfilled by the TOE.

O.Clean_Obj_Recovery (see 4.2.2). Object and data recovery free from malicious code

Recover to a viable state after malicious code is introduced and damage occurs, removing the malicious code as part of the process.

This objetive is assumed to be fulfilled by the environment.

O.Code_Signing (see 4.1.46). Code signing and verification

Check verification of signed downloaded code prior to execution. A well-known example is checking digital signatures on signed Java applets.

This objetive is fullfilled by the TOE.

O.General_Integ_Checks (see 4.1.67). Periodically check integrity

Provide periodic integrity checks on both system and user data.

This objetive is fullfilled by the TOE.

O.I&A_User_Action (see 4.1.74). User-action identification and authentication

Associate each user-requested action with the user who requested the action.

This objetive is fullfilled by the TOE.

O.Isolate_Executables (see 4.1.87). Isolate untrusted executables

Run executable code in a protected domain where the code's potential errors or malicious code will not significantly impact other system functions of other valid users of the system.

This objetive is fullfilled by the TOE.

O.Remote_Execution (see 4.1.109). Disable remote execution

Disable a remote entity's ability to execute local code.

This objetive is fullfilled by the TOE.

O.Trusted_Path (see 4.1.135). Provide a trusted path

Provide a trusted path between the user and the system. Execution of a user-requested action must be made via a trusted path with the following properties:

- The path is logically distinct from, and cannot be confused with other communication paths (by either the user or the system).
- The path provides assured identification of its end points.

This objetive is fullfilled by the TOE.

O.Trusted_Path&Channel (see 4.1.136). Trusted path and channel

Provide a trusted path to security-critical (TSF) data in which both end points have assured identities. For the remote user, there needs to be a trusted channel as well.

This objetive is fullfilled by the TOE.

7.2.57 Malicious code accidental IT download

DA.Mal_Code_IT_Download (see 3.6.6).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Clean_Obj_Recovery (see 4.2.2). Object and data recovery free from malicious code

Recover to a viable state after malicious code is introduced and damage occurs, removing the malicious code as part of the process.

This objetive is assumed to be fulfilled by the environment.

O.Code_Signing (see 4.1.46). Code signing and verification

Check verification of signed downloaded code prior to execution. A well-known example is checking digital signatures on signed Java applets.

This objetive is fullfilled by the TOE.

O.General_Integ_Checks (see 4.1.67). Periodically check integrity

Provide periodic integrity checks on both system and user data.

This objetive is fullfilled by the TOE.

O.Input_Inspection (see 4.1.78). Require inspection for absence of malicious code.

Require inspection of downloads/transfers.

This objetive is fullfilled by the TOE.

O.Obj_Protection (see 4.1.103). Object domain protection

Require domain protection for objects. Specify object classes (domains), user groups, and operation classes. Use these to specify which operations may be performed on which objects by which users. Basically this controls what users can do in a given group.

This objetive is fullfilled by the TOE.

7.2.58 Malicious code IT execution

DA.Mal_Code_IT_Exe (see 3.6.7).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Code_Val (see 4.1.6). Administrative validation of executables

Validate executable objects prior to allowing execution. Validation needs to be done by someone with an expertise to recognize malicious code and the authority and means to prevent its execution.

This objetive is fullfilled by the TOE.

O.Clean_Obj_Recovery (see 4.2.2). Object and data recovery free from malicious code

Recover to a viable state after malicious code is introduced and damage occurs, removing the malicious code as part of the process.

This objetive is assumed to be fulfilled by the environment.

O.Code_Signing (see 4.1.46). Code signing and verification

Check verification of signed downloaded code prior to execution. A well-known example is checking digital signatures on signed Java applets.

This objetive is fullfilled by the TOE.

O.General_Integ_Checks (see 4.1.67). Periodically check integrity

Provide periodic integrity checks on both system and user data.

This objetive is fullfilled by the TOE.

O.I&A_User_Action (see 4.1.74). User-action identification and authentication

Associate each user-requested action with the user who requested the action.

This objetive is fullfilled by the TOE.

O.Isolate_Executables (see 4.1.87). Isolate untrusted executables

Run executable code in a protected domain where the code's potential errors or malicious code will not significantly impact other system functions of other valid users of the system.

This objetive is fullfilled by the TOE.

7.2.59 Malicious code accidental user download

DA.Mal_Code_Usr_Downld (see 3.6.8).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Clean_Obj_Recovery (see 4.2.2). Object and data recovery free from malicious code

Recover to a viable state after malicious code is introduced and damage occurs, removing the malicious code as part of the process.

This objetive is assumed to be fulfilled by the environment.

O.Code_Signing (see 4.1.46). Code signing and verification

Check verification of signed downloaded code prior to execution. A well-known example is checking digital signatures on signed Java applets.

This objetive is fullfilled by the TOE.

O.Input_Inspection (see 4.1.78). Require inspection for absence of malicious code.

Require inspection of downloads/transfers.

This objetive is fullfilled by the TOE.

O.Obj_Protection (see 4.1.103). Object domain protection

Require domain protection for objects. Specify object classes (domains), user groups, and operation classes. Use these to specify which operations may be performed on which objects by which users. Basically this controls what users can do in a given group.

This objetive is fullfilled by the TOE.

7.2.60 Malicious code user execution

DA.Mal_Code_Usr_Exe (see 3.6.9).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Code_Val (see 4.1.6). Administrative validation of executables

Validate executable objects prior to allowing execution. Validation needs to be done by someone with an expertise to recognize malicious code and the authority and means to prevent its execution.

This objetive is fullfilled by the TOE.

O.Clean_Obj_Recovery (see 4.2.2). Object and data recovery free from malicious code

Recover to a viable state after malicious code is introduced and damage occurs, removing the malicious code as part of the process.

This objetive is assumed to be fulfilled by the environment.

O.Code_Signing (see 4.1.46). Code signing and verification

Check verification of signed downloaded code prior to execution. A well-known example is checking digital signatures on signed Java applets.

This objetive is fullfilled by the TOE.

O.General_Integ_Checks (see 4.1.67). Periodically check integrity

Provide periodic integrity checks on both system and user data.

This objetive is fullfilled by the TOE.

O.I&A_User_Action (see 4.1.74). User-action identification and authentication

Associate each user-requested action with the user who requested the action.

This objetive is fullfilled by the TOE.

O.Isolate_Executables (see 4.1.87). Isolate untrusted executables

Run executable code in a protected domain where the code's potential errors or malicious code will not significantly impact other system functions of other valid users of the system.

This objetive is fullfilled by the TOE.

7.2.61 Resource depletion failure

DA.Phys_CompFail_Res (see 3.4.42).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Priority_Of_Service (see 4.1.104). Provide priority of service

Control access to resources so that lower-priority activities do not unduly interfere with or delay higher-priority activities.

This objetive is fullfilled by the TOE.

O.Resource_Quotas (see 4.1.110). Resource quotas for users and services

Use resource quotas to limit user and service use of system resources to a level that will prevent degradation or denial of service to other critical users and services.

This objetive is fullfilled by the TOE.

7.2.62 Unexpected power reset

DA.Power_Disrupt_Reset (see 3.4.43).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Atomic_Functions (see 4.1.36). Complete security functions or recover to previous state

Recover automatically to a consistent, secure state if a security function does not complete successfully in the presence of certain types of failures.

This objetive is fullfilled by the TOE.

O.Trusted_Recovery (see 4.1.137). Trusted recovery of security functionality

Recovery to a secure state, without security compromise, after a discontinuity of operations.

This objetive is fullfilled by the TOE.

7.2.63 Denial of having received data from another local user

DA.Repudiate_Rcvr_Int (see 3.6.21).
The following Security Objective(s) are considered sufficient to counter this attack:

O.NonRepud_Locals_Rcvd (see 4.2.9). Non-repudiation for received information, local users

Prevent user from avoiding accountability for receiving a message from another user on the same system by providing evidence that the user received the message.

This objetive is assumed to be fulfilled by the environment.

7.2.64 Denial of having received information from a remote user

DA.Repudiate_Rcvr_Local (see 3.6.22).
The following Security Objective(s) are considered sufficient to counter this attack:

O.NonRepud_Assess_Recd (see 4.2.7). Non-repudiation support for received information by a nonlocal sender's TSF

Support nonrepudiation for received information by supporting remote handling of non-repudiation evidence if needed.

This objetive is assumed to be fulfilled by the environment.

O.NonRepud_Gen_Recd (see 4.2.8). Non-repudiation support for received information by the recipient's TSF

Prevent a receiving user from avoiding accountability for receiving a message by providing evidence that the user received the message.

This objetive is assumed to be fulfilled by the environment.

7.2.65 Denial of having received information by a remote user

DA.Repudiate_Rcvr_Rem (see 3.6.23).
The following Security Objective(s) are considered sufficient to counter this attack:

O.NonRepud_Assess_Recd (see 4.2.7). Non-repudiation support for received information by a nonlocal sender's TSF

Support nonrepudiation for received information by supporting remote handling of non-repudiation evidence if needed.

This objetive is assumed to be fulfilled by the environment.

O.NonRepud_Gen_Recd (see 4.2.8). Non-repudiation support for received information by the recipient's TSF

Prevent a receiving user from avoiding accountability for receiving a message by providing evidence that the user received the message.

This objetive is assumed to be fulfilled by the environment.

7.2.66 Denial of having sent information to another local user

DA.Repudiate_Send_Int (see 3.4.44).
The following Security Objective(s) are considered sufficient to counter this attack:

O.NonRepud_Locals_Sent (see 4.1.100). Non-repudiation for sent information, local users

Prevent user from avoiding accountability for sending a message to another user on the same system by providing evidence that the user sent the message.

This objetive is fullfilled by the TOE.

7.2.67 Denial of having sent information to a remote user

DA.Repudiate_Send_Local (see 3.4.45).
The following Security Objective(s) are considered sufficient to counter this attack:

O.NonRepud_Assess_Sent (see 4.1.98). Non-repudiation support for sent information by the nonlocal receiving TSF.

Support nonrepudiation for sent information by supporting remote handling of nonrepudiation evidence if needed.

This objetive is fullfilled by the TOE.

O.NonRepud_Gen_Sent (see 4.1.99). Non-repudiation support for sent information by the sender's TSF.

Prevent a user from avoiding accountability for sending a message to a recipient at a different site by providing evidence that the user sent the message.

This objetive is fullfilled by the TOE.

7.2.68 Denial of having sent data by a remote user

DA.Repudiate_Send_Rem (see 3.4.46).

The following Security Objective(s) are considered sufficient to counter this attack:

O.NonRepud_Assess_Sent (see 4.1.98). Non-repudiation support for sent information by the nonlocal receiving TSF.

Support nonrepudiation for sent information by supporting remote handling of nonrepudiation evidence if needed.

This objetive is fullfilled by the TOE.

O.NonRepud_Gen_Sent (see 4.1.99). Non-repudiation support for sent information by the sender's TSF.

Prevent a user from avoiding accountability for sending a message to a recipient at a different site by providing evidence that the user sent the message.

This objetive is fullfilled by the TOE.

7.2.69 Circumvent non-repudiation in a transaction involving a user and a local system

DA.Repudiate_Trans_Loc (see 3.6.24).

The following Security Objective(s) are considered sufficient to counter this attack:

O.I&A_Transaction (see 4.1.72). Transaction identification and authentication

Associate each transaction between a user and a system/application with a unique transaction ID, allowing events associated with a given transaction to be distinguished from other events involving the user and/or system/application.

This objetive is fullfilled by the TOE.

O.NonRepud_Locals_Rcvd (see 4.2.9). Non-repudiation for received information, local users

Prevent user from avoiding accountability for receiving a message from another user on the same system by providing evidence that the user received the message.

This objetive is assumed to be fulfilled by the environment.

O.NonRepud_Locals_Sent (see 4.1.100). Non-repudiation for sent information, local users

Prevent user from avoiding accountability for sending a message to another user on the same system by providing evidence that the user sent the message.

This objetive is fullfilled by the TOE.

7.2.70 Circumvent non-repudiation in a transaction involving a local user and a remote system

DA.Repudiate_Trans_Uloc (see 3.6.25).

The following Security Objective(s) are considered sufficient to counter this attack:

O.I&A_Transaction (see 4.1.72). Transaction identification and authentication

Associate each transaction between a user and a system/application with a unique transaction ID, allowing events associated with a given transaction to be distinguished from other events involving the user and/or system/application.

This objetive is fullfilled by the TOE.

O.NonRepud_Assess_Recd (see 4.2.7). Non-repudiation support for received information by a nonlocal sender's TSF

Support nonrepudiation for received information by supporting remote handling of non-repudiation evidence if needed.

This objetive is assumed to be fulfilled by the environment.

O.NonRepud_Assess_Sent (see 4.1.98). Non-repudiation support for sent information by the nonlocal receiving TSF.

Support nonrepudiation for sent information by supporting remote handling of nonrepudiation evidence if needed.

This objetive is fullfilled by the TOE.

O.NonRepud_Gen_Recd (see 4.2.8). Non-repudiation support for received information by the recipient's TSF

Prevent a receiving user from avoiding accountability for receiving a message by providing evidence that the user received the message.

This objetive is assumed to be fulfilled by the environment.

O.NonRepud_Gen_Sent (see 4.1.99). Non-repudiation support for sent information by the sender's TSF.

Prevent a user from avoiding accountability for sending a message to a recipient at a different site by providing evidence that the user sent the message.

This objetive is fullfilled by the TOE.

7.2.71 Circumvent non-repudiation in a transaction involving a remote user and a local system

DA.Repudiate_Trans_Urem (see 3.6.26).

The following Security Objective(s) are considered sufficient to counter this attack:

O.I&A_Transaction (see 4.1.72). Transaction identification and authentication

Associate each transaction between a user and a system/application with a unique transaction ID, allowing events associated with a given transaction to be distinguished from other events involving the user and/or system/application.

This objetive is fullfilled by the TOE.

O.NonRepud_Assess_Recd (see 4.2.7). Non-repudiation support for received information by a nonlocal sender's TSF

Support nonrepudiation for received information by supporting remote handling of non-repudiation evidence if needed.

This objetive is assumed to be fulfilled by the environment.

O.NonRepud_Assess_Sent (see 4.1.98). Non-repudiation support for sent information by the nonlocal receiving TSF.

Support nonrepudiation for sent information by supporting remote handling of nonrepudiation evidence if needed.

This objetive is fullfilled by the TOE.

O.NonRepud_Gen_Recd (see 4.2.8). Non-repudiation support for received information by the recipient's TSF

Prevent a receiving user from avoiding accountability for receiving a message by providing evidence that the user received the message.

This objetive is assumed to be fulfilled by the environment.

O.NonRepud_Gen_Sent (see 4.1.99). Non-repudiation support for sent information by the sender's TSF.

Prevent a user from avoiding accountability for sending a message to a recipient at a different site by providing evidence that the user sent the message.

This objetive is fullfilled by the TOE.

7.2.72 System use uncovers an intrinsic software flaw in a critical system component

DA.Software_Flaw (see 3.6.19).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Fail_Secure (see 4.1.66). Preservation of secure state for failures in critical components

Preserve the secure state of the system in the event of a secure component failure.

This objetive is fullfilled by the TOE.

O.Fault_Tolerance (see 4.2.4). Provide fault tolerant operations for critical components

Provide fault tolerant operations for critical components and continue to operate in the presence of specific failures in one or more system components.

This objetive is assumed to be fulfilled by the environment.

7.2.73 Accidental release of cryptographic assets due to TSF flaw or malfunction

DA.TSF_Err_Conf_Crypto (see 3.4.47).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Crypto_Data_Sep (see 4.1.52). Separation of cryptographic data

Provide complete separation between plaintext and encrypted data and between data and keys. This requires separate channels and separate storage areas. The only place any data can pass between the plaintext and encrypted data modules is in the cryptographic engine. There should be no way for plaintext keys to reach either data module and no way for data to enter the key handling module. Encrypted keys can be handled as encrypted data, but with limited user access.

This objetive is fullfilled by the TOE.

O.Crypto_Dsgn_Impl (see 4.1.53). Cryptographic Design and Implementation

Minimize or even eliminate design and implementation errors in the cryptographic modules and functions.

This objetive is fullfilled by the TOE.

O.Crypto_Key_Man (see 4.1.55). Cryptographic Key Management

Fully define cryptographic components, functions, and interfaces. Ensure appropriate protection for cryptographic keys throughout their lifecycle, covering generation, distribution, storage, use, and destruction.

This objetive is fullfilled by the TOE.

O.Crypto_Modular_Dsgn (see 4.1.57). Cryptographic Modular Design

Prevent errors in one part of the TOE from influencing other parts, especially cryptographic parts. To this end, noncryptographic I/O paths must be well defined and logically independent of circuitry and processes performing key generation, manual key entry, key zeroising, and similar key-related operations.

This objetive is fullfilled by the TOE.

O.Crypto_Operation (see 4.1.58). Cryptographic function definition

Cryptographic components, functions, and interfaces shall be fully defined.

This objetive is fullfilled by the TOE.

O.Crypto_Self_Test (see 4.1.59). Cryptographic self test

Provide the ability to verify that the cryptographic functions operate as designed.

This objetive is fullfilled by the TOE.

O.Crypto_Test_Reqs (see 4.1.60). Test cryptographic functionality

Test cryptographic operation and key management.

This objetive is fullfilled by the TOE.

O.Fail_Secure (see 4.1.66). Preservation of secure state for failures in critical components

Preserve the secure state of the system in the event of a secure component failure.

This objetive is fullfilled by the TOE.

O.Secure_State (see 4.1.113). Protect and maintain secure system state

Maintain and recover to a secure state without security compromise after system error or other interruption of system operation.

This objetive is fullfilled by the TOE.

7.2.74 User smuggles data using removable media

DA.User_Abuse_Conf_Disk (see 3.4.48).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Guidance (see 4.1.8). Administrator guidance documentation

Deter administrator errors by providing adequate administrator guidance.

This objetive is fullfilled by the TOE.

O.Audit_Account (see 4.1.38). Auditing for user accountability

Provide information about past user behavior to an authorized user through system mechanisms. Specifically, during any specified time interval, the system is able to report to a user acting in an identified audit role selected auditable actions that a user has performed, and as a result, what auditable objects were affected and what auditable information was received by that user.

This objetive is fullfilled by the TOE.

O.Data_Export_Control (see 4.1.62). Control user data exportation

Impose information control policies that do not allow export of specified data and/or export to specified locations.

This objetive is fullfilled by the TOE.

7.2.75 Steganographic data smuggling

DA.User_Abuse_Conf_Steg (see 3.4.49).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Admin_Code_Val (see 4.1.6). Administrative validation of executables

Validate executable objects prior to allowing execution. Validation needs to be done by someone with an expertise to recognize malicious code and the authority and means to prevent its execution.

This objetive is fullfilled by the TOE.

O.Admin_Code_Val_Sten (see 4.1.7). Software validation for absence of steganography

Validate exported objects for absence of steganographic content prior to allowing exportation. Validation needs to be done by someone with an expertise to recognize hidden content and the authority and means to prevent its export.

This objetive is fullfilled by the TOE.

O.Export_Control (see 4.1.64). Sanitize data objects containing hidden or unused data

Sanitize data objects that may contain hidden data when they are exported from the TOE in order to inhibit steganographic smuggling.

This objetive is fullfilled by the TOE.

7.2.76 User collects data by browsing

DA.User_Collect_Browse (see 3.4.50).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Account (see 4.1.38). Auditing for user accountability

Provide information about past user behavior to an authorized user through system mechanisms. Specifically, during any specified time interval, the system is able to report to a user acting in an identified audit role selected auditable actions that a user has performed, and as a result, what auditable objects were affected and what auditable information was received by that user.

This objetive is fullfilled by the TOE.

O.Info_Flow_Control (see 4.1.76). System enforced information flow

Enforce an information flow policy whereby users are constrained from allowing access to information they control, regardless of their intent (e.g., mandatory access control). This lattice property of security attributes is commonly associated with the U.S. DoD implementations of Mandatory Access Control (MAC).

This objetive is fullfilled by the TOE.

O.User_Defined_AC (see 4.1.145). User-defined access control

Enforce an access control policy whereby users may determine who may access information they control.

This objetive is fullfilled by the TOE.

7.2.77 User collects authentication data by deception

DA.User_Collect_Deceive (see 3.4.51).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Access_History (see 4.1.3). Access history for user session

Display information related to the most recent successful and unsuccessful attempts to establish a user session, once a user successfully establishes a user session.

This objetive is fullfilled by the TOE.

O.Trusted_Path (see 4.1.135). Provide a trusted path

Provide a trusted path between the user and the system. Execution of a user-requested action must be made via a trusted path with the following properties:

- The path is logically distinct from, and cannot be confused with other communication paths (by either the user or the system).
- The path provides assured identification of its end points.

This objetive is fullfilled by the TOE.

7.2.78 User collects data by deduction

DA.User_Collect_Deduce (see 3.4.52).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Generation (see 4.1.42). Audit records with identity

Record in audit records: date and time of action, location of the action, and the entity responsible for the action.

This objetive is fullfilled by the TOE.

O.Info_Flow_Control (see 4.1.76). System enforced information flow

Enforce an information flow policy whereby users are constrained from allowing access to information they control, regardless of their intent (e.g., mandatory access control). This lattice property of security attributes is commonly associated with the U.S. DoD implementations of Mandatory Access Control (MAC).

This objetive is fullfilled by the TOE.

O.User_Defined_AC (see 4.1.145). User-defined access control

Enforce an access control policy whereby users may determine who may access information they control.

This objetive is fullfilled by the TOE.

7.2.79 User collects data by eavesdropping

DA.User_Collect_Eaves (see 3.4.53).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Data_Exchange_Conf (see 4.1.61). Enforce data exchange confidentiality

Protect user data confidentiality when exchanging data with a remote system.

This objetive is fullfilled by the TOE.

O.Integ_User_Data_Int (see 4.1.82). Protect user data during internal transfer

Ensure the integrity of user data transferred internally within the system.

This objetive is fullfilled by the TOE.

O.Security_Roles (see 4.1.117). Security roles

Maintain security-relevant roles and the association of users with those roles.

This objetive is fullfilled by the TOE.

7.2.80 User collects residual data

DA.User_Collect_Residue (see 3.4.54).
The following Security Objective(s) are considered sufficient to counter this attack:

O.No_Residual_Info (see 4.1.97). Eliminate residual information

Ensure there is no object reuse; i.e., ensure that there is no residual information in some information containers or system resources upon their reallocation to different users.

This objetive is fullfilled by the TOE.

7.2.81 User's unauthorized use causes overload of communication resources

DA.User_Comm_Overload (see 3.4.55).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Generation (see 4.1.42). Audit records with identity

Record in audit records: date and time of action, location of the action, and the entity responsible for the action.

This objetive is fullfilled by the TOE.

O.Data_Imp_Exp_Control (see 4.1.63). Data import/export to/from system control

Protect data from being sent to erroneous places and more places external to the system than allowed by the organization's security policy. Conversely the import of data into the system should be protected from illicit information or information not allowed by the organization's security policy.

This objetive is fullfilled by the TOE.

O.Limit_Comm_Sessions (see 4.1.90). Limit the number of user initiated communication sessions

Provide mechanisms to limit the number of sessions that the user can initiate, if the user initiates multiple sessions that exceed the processors ability to perform in a reliable and efficient manner. These sessions could ether be communication (TCP/IP) sessions or user login sessions.

This objetive is fullfilled by the TOE.

O.Priority_Of_Service (see 4.1.104). Provide priority of service

Control access to resources so that lower-priority activities do not unduly interfere with or delay higher-priority activities.

This objetive is fullfilled by the TOE.

O.Resource_Quotas (see 4.1.110). Resource quotas for users and services

Use resource quotas to limit user and service use of system resources to a level that will prevent degradation or denial of service to other critical users and services.

This objetive is fullfilled by the TOE.

7.2.82 Denial of service due to exhausted audit storage

DA.User_ErrAvl_AudExhst (see 3.4.56).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Loss_Respond (see 4.1.43). Respond to possible loss of stored audit records

Respond to possible loss of audit records when audit trail storage is full or nearly full.

This objetive is fullfilled by the TOE.

O.Guarantee_Audit_Stg (see 4.1.68). Guarantee the availability of audit storage space

Maintain audit data and guarantee space for that data.

This objetive is fullfilled by the TOE.

O.Manage_TSF_Data (see 4.1.96). Manage security-critical data to avoid storage space being exceeded

Manage security-critical (TSF) data to ensure that the size of the data does not exceed the space allocated for storage of the data.

This objetive is fullfilled by the TOE.

7.2.83 Falsification of information quality in data export

DA.User_Err_AttrXpt (see 3.4.57).

The following Security Objective(s) are considered sufficient to counter this attack:

O.AC_Label_Export (see 4.1.2). Object security attributes and exportation

Provide object security attributes in exported data with moderate to high effectiveness. The attributes are those associated with specific security function policies.

This objetive is fullfilled by the TOE.

7.2.84 Under-classification of data sensitivity on export

DA.User_Err_Conf_Class (see 3.4.58).
The following Security Objective(s) are considered sufficient to counter this attack:

O.AC_Label_Export (see 4.1.2). Object security attributes and exportation

Provide object security attributes in exported data with moderate to high effectiveness. The attributes are those associated with specific security function policies.

This objetive is fullfilled by the TOE.

7.2.85 Accidental release of cryptographic assets due to user error

DA.User_Err_Conf_Crypto (see 3.4.59).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Crypto_AC (see 4.1.50). Cryptographic access control policy

Restrict user access to cryptographic IT assets in accordance with a specified user access control policy.

This objetive is fullfilled by the TOE.

O.Crypto_Key_Man (see 4.1.55). Cryptographic Key Management

Fully define cryptographic components, functions, and interfaces. Ensure appropriate protection for cryptographic keys throughout their lifecycle, covering generation, distribution, storage, use, and destruction.

This objetive is fullfilled by the TOE.

O.I&A_Domain (see 4.1.71). Identify and authenticate a user to support accountability

Provide the basic I&A functions that will support user accountability.

This objetive is fullfilled by the TOE.

O.I&A_User_Action (see 4.1.74). User-action identification and authentication

Associate each user-requested action with the user who requested the action.

This objetive is fullfilled by the TOE.

7.2.86 Confidentiality violation of export control policy

DA.User_Err_Conf_Exp (see 3.4.60).
The following Security Objective(s) are considered sufficient to counter this attack:

O.User_Conf_Prevention (see 4.1.141). Basic confidentiality-breach prevention

Prevent unauthorized export of confidential information from the TOE with moderate effectiveness.

This objetive is fullfilled by the TOE.

7.2.87 User error deletes data

DA.User_Err_Delete (see 3.6.27).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Rollback (see 4.2.17). Rollback

Recover from user operations by undoing some user operations (i.e., rolling back) to restore a previous known state.

This objetive is assumed to be fulfilled by the environment.

O.User_Guidance (see 4.1.146). User guidance documentation

Provide documentation for the general user.

This objetive is fullfilled by the TOE.

7.2.88 User error modifying attributes availability

DA.User_Err_Mod_Attr (see 3.4.61).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Security_Attr_Mgt (see 4.1.114). Manage security attributes

Manage the initialization of, values for, and allowable operations on security attributes.

This objetive is fullfilled by the TOE.

O.User_Guidance (see 4.1.146). User guidance documentation

Provide documentation for the general user.

This objetive is fullfilled by the TOE.

7.2.89 Failure to provide object security attributes in data export

DA.User_Err_MsngAttrXpt (see 3.4.62).
The following Security Objective(s) are considered sufficient to counter this attack:

O.AC_Label_Export (see 4.1.2). Object security attributes and exportation

Provide object security attributes in exported data with moderate to high effectiveness. The attributes are those associated with specific security function policies.

This objetive is fullfilled by the TOE.

7.2.90 Incorrectly set object attributes

DA.User_Err_Object_Attr (see 3.4.63).
The following Security Objective(s) are considered sufficient to counter this attack:

O.AC_Label_Export (see 4.1.2). Object security attributes and exportation

Provide object security attributes in exported data with moderate to high effectiveness. The attributes are those associated with specific security function policies.

This objetive is fullfilled by the TOE.

O.Obj_Attr_Integrity (see 4.1.102). Basic object attribute integrity

Maintain object security attributes with moderate to high accuracy (under the guidance of qualified users).

This objetive is fullfilled by the TOE.

7.2.91 User error setting attributes availability

DA.User_Err_Set_Attr (see 3.4.64).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Security_Attr_Mgt (see 4.1.114). Manage security attributes

Manage the initialization of, values for, and allowable operations on security attributes.

This objetive is fullfilled by the TOE.

O.User_Guidance (see 4.1.146). User guidance documentation

Provide documentation for the general user.

This objetive is fullfilled by the TOE.

7.2.92 User modifies audit trail

DA.User_Modify_Audit (see 3.4.65).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Gen_User (see 4.1.41). Individual accountability

Provide individual accountability for audited events. Uniquely identify each user so that auditable actions can be traced to a user.

This objetive is fullfilled by the TOE.

O.Audit_Generation (see 4.1.42). Audit records with identity

Record in audit records: date and time of action, location of the action, and the entity responsible for the action.

This objetive is fullfilled by the TOE.

O.Audit_Protect (see 4.1.44). Protect stored audit records

Protect audit records against unauthorized access, modification, or deletion to ensure accountability of user actions.

This objetive is fullfilled by the TOE.

O.Security_Roles (see 4.1.117). Security roles

Maintain security-relevant roles and the association of users with those roles.

This objetive is fullfilled by the TOE.

7.2.93 User improperly modifies authentication data

DA.User_Modify_Auth (see 3.4.66).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Account (see 4.1.38). Auditing for user accountability

Provide information about past user behavior to an authorized user through system mechanisms. Specifically, during any specified time interval, the system is able to report to a user acting in an identified audit role selected auditable actions that a user has performed, and as a result, what auditable objects were affected and what auditable information was received by that user.

This objetive is fullfilled by the TOE.

O.Security_Data_Mgt (see 4.1.115). Manage security-critical data

Manage the initialization of, limits on, and allowable operations on security-critical data.

This objetive is fullfilled by the TOE.

7.2.94 User improperly modifies user data

DA.User_Modify_Data (see 3.4.67).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Generation (see 4.1.42). Audit records with identity

Record in audit records: date and time of action, location of the action, and the entity responsible for the action.

This objetive is fullfilled by the TOE.

O.Info_Flow_Control (see 4.1.76). System enforced information flow

Enforce an information flow policy whereby users are constrained from allowing access to information they control, regardless of their intent (e.g., mandatory access control). This lattice property of security attributes is commonly associated with the U.S. DoD implementations of Mandatory Access Control (MAC).

This objetive is fullfilled by the TOE.

O.User_Defined_AC (see 4.1.145). User-defined access control

Enforce an access control policy whereby users may determine who may access information they control.

This objetive is fullfilled by the TOE.

7.2.95 User improperly modifies TSF data

DA.User_Modify_TSFData (see 3.4.68).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Generation (see 4.1.42). Audit records with identity

Record in audit records: date and time of action, location of the action, and the entity responsible for the action.

This objetive is fullfilled by the TOE.

O.Config_Management (see 4.1.48). Implement operational configuration management

Implement a configuration management plan. Implement configuration management to assure storage integrity, identification of system connectivity (software, hardware, and firmware), and identification of system components (software, hardware, and firmware).

This objetive is fullfilled by the TOE.

O.General_Integ_Checks (see 4.1.67). Periodically check integrity

Provide periodic integrity checks on both system and user data.

This objetive is fullfilled by the TOE.

O.Info_Flow_Control (see 4.1.76). System enforced information flow

Enforce an information flow policy whereby users are constrained from allowing access to information they control, regardless of their intent (e.g., mandatory access control). This lattice property of security attributes is commonly associated with the U.S. DoD implementations of Mandatory Access Control (MAC).

This objetive is fullfilled by the TOE.

O.Integ_Sys_Data_Int (see 4.1.81). Integrity of system data transferred internally

Ensure the integrity of system data transferred internally.

This objetive is fullfilled by the TOE.

O.Integrity_Practice (see 4.1.86). Operational integrity system function testing

Provide system functional tests to periodically test the integrity of the hardware and code running system functions.

This objetive is fullfilled by the TOE.

O.Maintain_Sec_Domain (see 4.1.92). Maintain security domain

Maintain at least one security domain for system (TOE) execution to protect the TOE from interference and tampering.

This objetive is fullfilled by the TOE.

O.Reference_Monitor (see 4.1.108). Provide reference monitor

Always invoke mechanisms that enforce security policies (i.e., as for a traditional reference monitor).

This objetive is fullfilled by the TOE.

O.User_Defined_AC (see 4.1.145). User-defined access control

Enforce an access control policy whereby users may determine who may access information they control.

This objetive is fullfilled by the TOE.

7.2.96 User obstructs legitimate use of resources.

DA.User_Obst_Res_Use (see 3.6.28).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Manage_Res_Sec_Attr (see 4.1.95). Manage resource security attributes

Provide management on resource security attributes.

This objetive is fullfilled by the TOE.

O.Priority_Of_Service (see 4.1.104). Provide priority of service

Control access to resources so that lower-priority activities do not unduly interfere with or delay higher-priority activities.

This objetive is fullfilled by the TOE.

O.Tamper_ID (see 4.2.19). Tamper detection

Provide system features that detect physical tampering of a system component, and use those features to limit security breaches.

This objetive is assumed to be fulfilled by the environment.

O.Tamper_Resistance (see 4.2.20). Tamper resistance

Prevent or resist physical tampering with specified system devices and components.

This objetive is assumed to be fulfilled by the environment.

7.2.97 User's unauthorized actions over-task the system causing processor overload

DA.User_Prcsr_Overload (see 3.4.69).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Generation (see 4.1.42). Audit records with identity

Record in audit records: date and time of action, location of the action, and the entity responsible for the action.

This objetive is fullfilled by the TOE.

O.Limit_Comm_Sessions (see 4.1.90). Limit the number of user initiated communication sessions

Provide mechanisms to limit the number of sessions that the user can initiate, if the user initiates multiple sessions that exceed the processors ability to perform in a reliable and efficient manner. These sessions could ether be communication (TCP/IP) sessions or user login sessions.

This objetive is fullfilled by the TOE.

O.Priority_Of_Service (see 4.1.104). Provide priority of service

Control access to resources so that lower-priority activities do not unduly interfere with or delay higher-priority activities.

This objetive is fullfilled by the TOE.

O.Resource_Quotas (see 4.1.110). Resource quotas for users and services

Use resource quotas to limit user and service use of system resources to a level that will prevent degradation or denial of service to other critical users and services.

This objetive is fullfilled by the TOE.

7.2.98 User sends data violating confidentiality

DA.User_Send_Conf (see 3.4.70).
The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Generation (see 4.1.42). Audit records with identity

Record in audit records: date and time of action, location of the action, and the entity responsible for the action.

This objetive is fullfilled by the TOE.

O.Integ_Data_Mark_Exp (see 4.1.79). Data marking integrity export

Ensure that data markings are included with data that is exported to another trusted product.

This objetive is fullfilled by the TOE.

7.2.99 User sends data violating integrity

DA.User_Send_Integrity (see 3.4.71).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Generation (see 4.1.42). Audit records with identity

Record in audit records: date and time of action, location of the action, and the entity responsible for the action.

This objetive is fullfilled by the TOE.

O.Integ_Data_Mark_Exp (see 4.1.79). Data marking integrity export

Ensure that data markings are included with data that is exported to another trusted product.

This objetive is fullfilled by the TOE.

7.2.100 User's unauthorized actions cause storage overload

DA.User_Stg_Overload (see 3.4.72).

The following Security Objective(s) are considered sufficient to counter this attack:

O.Audit_Generation (see 4.1.42). Audit records with identity

Record in audit records: date and time of action, location of the action, and the entity responsible for the action.

This objetive is fullfilled by the TOE.

O.Limit_Comm_Sessions (see 4.1.90). Limit the number of user initiated communication sessions

Provide mechanisms to limit the number of sessions that the user can initiate, if the user initiates multiple sessions that exceed the processors ability to perform in a reliable and efficient manner. These sessions could ether be communication (TCP/IP) sessions or user login sessions.

This objetive is fullfilled by the TOE.

O.Priority_Of_Service (see 4.1.104). Provide priority of service

Control access to resources so that lower-priority activities do not unduly interfere with or delay higher-priority activities.

This objetive is fullfilled by the TOE.

O.Resource_Quotas (see 4.1.110). Resource quotas for users and services

Use resource quotas to limit user and service use of system resources to a level that will prevent degradation or denial of service to other critical users and services.

This objetive is fullfilled by the TOE.

7.3 Detailed Policy Statements - General Policy Statement Mapping

To complete the Security Policy Statement analysis, this section presents an overall mapping of Detailed Policy Statements with respect to their parent General Policy Statements.

This mapping provides a coverage analysis, where for each General Policy Statement, its components in terms of Detailed Policy Statements are shown, and then it can be proven that all of them are covered, either by the TOE or by the Environment, by application of the Assumptions.

7.3.1 Individual accountability

GP.Accountability (see 3.7.1). This General Policy Statement is divided into the following Detailed Policy Statement(s):

DP.Audit_Gen_User (see 3.8.9). Individual accountability

The system shall provide individual accountability for auditable actions. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Audit_Generation (see 3.8.10). Audit data generation with identity

The system shall provide the capability to ensure that all audit records include enough information to determine the date and time of action, the system locale of the action, the system entity that initiated or completed the action, the resources involved, and the action involved. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Audit_Protect (see 3.8.11). Protected audit data storage

The system shall protect the contents of the audit trails against unauthorized access, modification, or deletion. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.I&A_User (see 3.8.17). User identification and authentication

The system shall provide Identification and authentication procedures which uniquely identify and authenticate users. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.User_Defined_AC (see 3.8.35). Discretionary access control

The system shall provide a Discretionary Access Control (DAC) function (i.e., a user can grant access authorization to other users for data they control). This Detailed Policy Statement is fullfilled solely by the TOE.

7.3.2 Notification of threats and vulnerabilities

GP.Authorities (see 3.7.2). This General Policy Statement is divided into the following Detailed Policy Statement(s):

DP.Authority_Notify (see 3.8.12). Notification of threats and vulnerabilities

Notification of threats and vulnerabilities shall be addressed. This Detailed Policy Statement is fullfilled solely by the TOE.

7.3.3 Authorized use of information

GP.Authorized_Use (see 3.7.3). This General Policy Statement is divided into the following Detailed Policy Statement(s):

DP.Sys_Access_Banners (see 3.8.25). System access banners

The system shall notify users prior to gaining access that the user's actions may be monitored and recorded, that using the system consents to such monitoring, and that unauthorized use may result in criminal or civil penalties. This Detailed Policy Statement is fullfilled solely by the TOE.

7.3.4 Installation and usage guidance

GP.Guidance (see 3.7.4). This General Policy Statement is divided into the following Detailed Policy Statement(s):

DP.Privileged_Doc (see 3.8.22). Privileged user documentation

Documentation shall include guides or manuals for the system's privileged users. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.User_Documentation (see 3.8.36). General user documentation

Documentation shall include a user's guide for the general user. This Detailed Policy Statement is fullfilled solely by the TOE.

7.3.5 System lifecycle phases integrate security

GP.Lifecycle (see 3.7.5). This General Policy Statement is divided into the following Detailed Policy Statement(s):

DP.Lifecycle_Security (see 3.8.20). Security throughout lifecycle

Security shall be addressed throughout the system's lifecycle. This Detailed Policy Statement is fullfilled solely by the TOE.

7.3.6 Information marking

GP.Marking (see 3.7.6). This General Policy Statement is divided into the following Detailed Policy Statement(s):

DP.Config_Mgt_Plan (see 3.8.14). Implement operational configuration management

A configuration management plan shall be implemented by the system. The system shall implement configuration management to assure storage integrity, identification of system connectivity (software, hardware, and firmware), and identification of system components (software, hardware, and firmware). The system shall implement strong integrity mechanisms (integrity locks, encryption). This Detailed Policy Statement is fullfilled solely by the TOE.

DP.External_Labels (see 3.8.16). Labeling data

The system shall provide security parameters associated with information exchanged between systems. This Detailed Policy Statement is fullfilled solely by the TOE.

7.3.7 Semiformally designed, tested and reviewed

GP.Product_Assurance (see 3.7.7). This General Policy Statement is divided into the following Detailed Policy Statement(s):

DP.Assurance_ACM (see 3.8.2). Configuration Management

Configuration management (CM) helps to ensure that the integrity of the TOE is preserved, by requiring discipline and control in the processes of refinement and modification of the TOE and other related information. CM prevents unauthorised modifications, additions, or deletions to the TOE, thus providing assurance that the TOE and documentation used for evaluation are the ones prepared for distribution. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Assurance_ADO (see 3.8.3). Delivery and Operation

This security policy defines requirements for the measures, procedures, and standards concerned with secure delivery, installation, and operational use of the TOE, ensuring that the security protection offered by the TOE is not compromised during transfer, installation, start-up, and operation. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Assurance_ADV (see 3.8.4). Product Development

This security policy defines requirements for the stepwise refinement of the TSF from the TOE summary specification in the ST down to the actual implementation. Each of the resulting TSF representations provide information to help the evaluator determine whether the functional requirements of the TOE have been met. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Assurance_AGD (see 3.8.5). Guidance documents

This security policy defines requirements directed at the understandability, coverage and completeness of the operational documentation provided by the developer. This documentation, which provides two categories of information, for users and for administrators, is an important factor in the secure operation of the TOE. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Assurance_ALC (see 3.8.6). Lifecycle support

This security policy defines requirements for assurance through the adoption of a well defined life-cycle model for all the steps of the TOE development, including flaw remediation procedures and policies, correct use of tools and techniques and the security measures used to protect the development environment. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Assurance_ATE (see 3.8.7). Test

This security policy states testing requirements that demonstrate that the TSF satisfies the TOE security functional requirements. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Assurance_AVA (see 3.8.8). Vulnerability Assesment

This security policy defines requirements directed at the identification of exploitable vulnerabilities. Specifically, it addresses those vulnerabilities introduced in the construction, operation, misuse, or incorrect configuration of the TOE. This Detailed Policy Statement is fullfilled solely by the TOE.

7.3.8 Information availability

GP.Availability (see 3.9.1). This General Policy Statement is divided into the following Detailed Policy Statement(s):

DP.Config_Mgt_Plan (see 3.8.14). Implement operational configuration management

A configuration management plan shall be implemented by the system. The system shall implement configuration management to assure storage integrity, identification of system connectivity (software, hardware, and firmware), and identification of system components (software, hardware, and firmware). The system shall implement strong integrity mechanisms (integrity locks, encryption). This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Documented_Recovery (see 3.8.15). Documented recovery

The system shall provide procedures and features to assure that system recovery is done in a trusted and secure manner. Any circumstances that could result in an untrusted recovery shall be documented. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Malicious_Code (see 3.10.1). Malicious code prevention

Procedures and mechanisms to prevent the introduction of malicious code into the system shall be provided. This Detailed Policy Statement is responsibility of the Environment.

DP.Sys_Assur_HW/SW/FW (see 3.8.26). Validation of security function integrity

Features and procedures to validate the integrity and the expected operation of the security-relevant software, hardware, and firmware shall be provided by the system. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Sys_Backup_Procs (see 3.8.27). System backup procedures

Provide the capability to restore the system to a secure state after discontinuities of system operations. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Sys_Backup_Restore (see 3.8.28). Restoration with minimal loss

The system shall provide backup procedures to allow restoration of the system with minimal loss of service or data. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Sys_Backup_Storage (see 3.8.29). Effective backup restoration

The system shall provide procedures to ensure both the existence of sufficient backup storage capability and effective restoration (incremental and complete) of the backup data. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Sys_Backup_Verify (see 3.10.2). Backup protection and restoration

The system shall provide appropriate physical and technical protection of the backup and restoration hardware, firmware, and software. This Detailed Policy Statement is responsibility of the Environment.

DP.System_Recovery (see 3.8.31). Trusted system recovery

Provide procedures and features to assure that system recovery is done in a trusted and secure manner. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.User_Data_Dial-in (see 3.8.32). Encryption of transmitted user data

The system shall provide data transmission using an encryption mechanism appropriate for the sensitivity of the data. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.User_Data_Storage (see 3.8.33). Protection of stored user data

The system shall provide appropriate storage, continuous personnel access control storage, or encrypted storage of data based on the sensitivity of the data. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.User_Data_Transfer (see 3.8.34). Protection of transmitted user data

The system shall provide a protected distribution system for data transmitted. This Detailed Policy Statement is fullfilled solely by the TOE.

7.3.9 Information access control

GP.Information_AC (see 3.9.2). This General Policy Statement is divided into the following Detailed Policy Statement(s):

DP.Admin_Security_Data (see 3.8.1). Changes to security data by authorized personnel

Provide mechanisms to assure that changes to security related data are executed only by authorized personnel. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Need_To_Know (see 3.8.21). Privileged user access

The system shall function so that each user has access to all of the information and functions that the user requires to perform duties, but no more. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Screen_Lock (see 3.8.23). User screen locking

The system shall provide a screen lock mechanism. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.User_Auth_Enhanced (see 3.10.3). Enhanced user identification and authentication

The system shall require the use of enhanced authentication for privileged users who either reside outside of the system's perimeter or whose communications traverse data lines outside of the system's perimeter. This Detailed Policy Statement is responsibility of the Environment.

DP.User_Defined_AC (see 3.8.35). Discretionary access control

The system shall provide a Discretionary Access Control (DAC) function (i.e., a user can grant access authorization to other users for data they control). This Detailed Policy Statement is fullfilled solely by the TOE.

7.3.10 Information content integrity

GP.Integrity (see 3.9.3). This General Policy Statement is divided into the following Detailed Policy Statement(s):

DP.Admin_Security_Data (see 3.8.1). Changes to security data by authorized personnel

Provide mechanisms to assure that changes to security related data are executed only by authorized personnel. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Change_Control_Users (see 3.8.13). Notification of data content changes

Notify user of the time and date of the last modification of data. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Config_Mgt_Plan (see 3.8.14). Implement operational configuration management

A configuration management plan shall be implemented by the system. The system shall implement configuration management to assure storage integrity, identification of system connectivity (software, hardware, and firmware), and identification of system components (software, hardware, and firmware). The system shall implement strong integrity mechanisms (integrity locks, encryption). This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Documented_Recovery (see 3.8.15). Documented recovery

The system shall provide procedures and features to assure that system recovery is done in a trusted and secure manner. Any circumstances that could result in an untrusted recovery shall be documented. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Integrity_Data/SW (see 3.8.18). Strong integrity mechanisms

The system shall implement strong integrity mechanisms (integrity locks, encryption). This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Integrity_Practice (see 3.8.19). Operational integrity system function testing

Provide system functional tests to periodically test the integrity of the hardware and code running system functions. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Malicious_Code (see 3.10.1). Malicious code prevention

Procedures and mechanisms to prevent the introduction of malicious code into the system shall be provided. This Detailed Policy Statement is responsibility of the Environment.

DP.Non-Repudiation (see 3.10.4). Non-repudiation capabilities

The system shall provide non-repudiation capabilities. This Detailed Policy Statement is responsibility of the Environment.

DP.Storage_Integrity (see 3.8.24). Assurance of effective storage integrity

The system shall provide assurance that storage integrity is effective. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.Sys_Assur_HW/SW/FW (see 3.8.26). Validation of security function integrity

Features and procedures to validate the integrity and the expected operation of the security-relevant software, hardware, and firmware shall be provided by the system. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.System_Protection (see 3.8.30). Protection from security function modification

Provide features or procedures for protection of the system from improper changes. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.System_Recovery (see 3.8.31). Trusted system recovery

Provide procedures and features to assure that system recovery is done in a trusted and secure manner. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.User_Data_Dial-in (see 3.8.32). Encryption of transmitted user data

The system shall provide data transmission using an encryption mechanism appropriate for the sensitivity of the data. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.User_Data_Storage (see 3.8.33). Protection of stored user data

The system shall provide appropriate storage, continuous personnel access control storage, or encrypted storage of data based on the sensitivity of the data. This Detailed Policy Statement is fullfilled solely by the TOE.

DP.User_Data_Transfer (see 3.8.34). Protection of transmitted user data

The system shall provide a protected distribution system for data transmitted. This Detailed Policy Statement is fullfilled solely by the TOE.

7.3.11 Physical protection

GP.Physical_Control (see 3.9.4). This General Policy Statement is divided into the following Detailed Policy Statement(s):

DP.Tamper_ID (see 3.10.5). Physical tampering detection and notification

The system shall detect physical tampering and notify the appropriate authority. This Detailed Policy Statement is responsibility of the Environment.

7.4 Detailed Policy Statements - Security Objective Mapping

Those Detailed Security Objectives identified in Sections 3.8 and 3.8 are herein assigned to Security Objectives, that are effective and sufficient to comply with the terms of the policy. It can be shown that every Policy Statement is covered by such Security Requirements.

7.4.1 Changes to security data by authorized personnel

DP.Admin_Security_Data (see 3.8.1). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Security_Attr_Mgt (see 4.1.114). Manage security attributes This Security Objective is fullfilled by the TOE.

O.Security_Data_Mgt (see 4.1.115). Manage security-critical data This Security Objective is fullfilled by the TOE.

O.Security_Func_Mgt (see 4.1.116). Manage behavior of security functions This Security Objective is fullfilled by the TOE.

7.4.2 Configuration Management

DP.Assurance_ACM (see 3.8.2). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Assurance_ACM_AUT (see 4.1.10). CM automation This Security Objective is fullfilled by the TOE.

O.Assurance_ACM_CAP (see 4.1.11). CM capabilities This Security Objective is fullfilled by the TOE.

O.Assurance_ACM_SCP (see 4.1.12). CM scope This Security Objective is fullfilled by the TOE.

7.4.3 Delivery and Operation

DP.Assurance_ADO (see 3.8.3). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Assurance_ADO_DEL (see 4.1.13). Delivery This Security Objective is fullfilled by the TOE.

O.Assurance_ADO_IGS (see 4.1.14). Installation, generation and start-up This Security Objective is fullfilled by the TOE.

7.4.4 Product Development

DP.Assurance_ADV (see 3.8.4). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Assurance_ADV_FSP (see 4.1.15). Functional specification This Security Objective is fullfilled by the TOE.

O.Assurance_ADV_HLD (see 4.1.16). High-level design This Security Objective is fullfilled by the TOE.

O.Assurance_ADV_IMP (see 4.1.17). Implementation representation This Security Objective is fullfilled by the TOE.

O.Assurance_ADV_INT (see 4.1.18). TSF internals This Security Objective is fullfilled by the TOE.

O.Assurance_ADV_LLD (see 4.1.19). Low-level design This Security Objective is fullfilled by the TOE.

O.Assurance_ADV_RCR (see 4.1.20). Representation correspondence This Security Objective is fullfilled by the TOE.

O.Assurance_ADV_SPM (see 4.1.21). Security policy modeling This Security Objective is fullfilled by the TOE.

7.4.5 Guidance documents

DP.Assurance_AGD (see 3.8.5). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Assurance_AGD_ADM (see 4.1.22). Administrator guidance This Security Objective is fullfilled by the TOE.

O.Assurance_AGD_USR (see 4.1.23). User guidance This Security Objective is fullfilled by the TOE.

7.4.6 Lifecycle support

DP.Assurance_ALC (see 3.8.6). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Assurance_ALC_DVS (see 4.1.24). Development security This Security Objective is fullfilled by the TOE.

O.Assurance_ALC_FLR (see 4.1.25). Flaw remediation This Security Objective is fullfilled by the TOE.

O.Assurance_ALC_LCD (see 4.1.26). Life cycle definition This Security Objective is fullfilled by the TOE.

O.Assurance_ALC_TAT (see 4.1.27). Tools and techniques This Security Objective is fullfilled by the TOE.

7.4.7 Test

DP.Assurance_ATE (see 3.8.7). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Assurance_ATE_COV (see 4.1.28). Coverage This Security Objective is fullfilled by the TOE.

O.Assurance_ATE_DPT (see 4.1.29). Depth This Security Objective is fullfilled by the TOE.

O.Assurance_ATE_FUN (see 4.1.30). Functional tests This Security Objective is fullfilled by the TOE.

O.Assurance_ATE_IND (see 4.1.31). Independent testing This Security Objective is fullfilled by the TOE.

7.4.8 Vulnerability Assesment

DP.Assurance_AVA (see 3.8.8). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Assurance_AVA_CCA (see 4.1.32). Covert channel analysis This Security Objective is fullfilled by the TOE.

O.Assurance_AVA_MSU (see 4.1.33). Misuse This Security Objective is fullfilled by the TOE.

O.Assurance_AVA_SOF (see 4.1.34). Strength of TOE security functions This Security Objective is fullfilled by the TOE.

O.Assurance_AVA_VLA (see 4.1.35). Vulnerability analysis This Security Objective is fullfilled by the TOE.

7.4.9 Individual accountability

DP.Audit_Gen_User (see 3.8.9). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Audit_Gen_User (see 4.1.41). Individual accountability This Security Objective is fullfilled by the TOE.

7.4.10 Audit data generation with identity

DP.Audit_Generation (see 3.8.10). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Audit_Generation (see 4.1.42). Audit records with identity This Security Objective is fullfilled by the TOE.

7.4.11 Protected audit data storage

DP.Audit_Protect (see 3.8.11). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Audit_Protect (see 4.1.44). Protect stored audit records This Security Objective is fullfilled by the TOE.

7.4.12 Notification of threats and vulnerabilities

DP.Authority_Notify (see 3.8.12). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Admin_Guidance (see 4.1.8). Administrator guidance documentation This Security Objective is fullfilled by the TOE.

O.User_Guidance (see 4.1.146). User guidance documentation This Security Objective is fullfilled by the TOE.

7.4.13 Notification of data content changes

DP.Change_Control_Users (see 3.8.13). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Change_Control_Users (see 4.1.45). User notification of data content changes This Security Objective is fullfilled by the TOE.

7.4.14 Implement operational configuration management

DP.Config_Mgt_Plan (see 3.8.14). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Config_Management (see 4.1.48). Implement operational configuration management This Security Objective is fullfilled by the TOE.

7.4.15 Documented recovery

DP.Documented_Recovery (see 3.8.15). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Trusted_Recovery_Doc (see 4.1.138). Documentation of untrusted data recovery This Security Objective is fullfilled by the TOE.

7.4.16 Labeling data

DP.External_Labels (see 3.8.16). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.External_Labels (see 4.1.65). Label or mark information for external systems This Security Objective is fullfilled by the TOE.

7.4.17 User identification and authentication

DP.I&A_User (see 3.8.17). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.I&A_Domain (see 4.1.71). Identify and authenticate a user to support accountability This Security Objective is fullfilled by the TOE.

7.4.18 Strong integrity mechanisms

DP.Integrity_Data/SW (see 3.8.18). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Integrity_Data/SW (see 4.1.84). Integrity protection for user data and software This Security Objective is fullfilled by the TOE.

7.4.19 Operational integrity system function testing

DP.Integrity_Practice (see 3.8.19). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Integrity_Practice (see 4.1.86). Operational integrity system function testing This Security Objective is fullfilled by the TOE.

7.4.20 Security throughout lifecycle

DP.Lifecycle_Security (see 3.8.20). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Lifecycle_Security (see 4.1.88). Lifecycle security This Security Objective is fullfilled by the TOE.

7.4.21 Privileged user access

DP.Need_To_Know (see 3.8.21). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.User_Defined_AC (see 4.1.145). User-defined access control This Security Objective is fullfilled by the TOE.

The SFP gives object owners the ability to restrict direct access to those with a designated need to know. However, those with a need-to-know (and processes acting on their behalf) must be trusted to themselves enforce need-to-know constraints.

7.4.22 Privileged user documentation

DP.Privileged_Doc (see 3.8.22). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Admin_Guidance (see 4.1.8). Administrator guidance documentation This Security Objective is fullfilled by the TOE.

7.4.23 User screen locking

DP.Screen_Lock (see 3.8.23). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Screen_Lock (see 4.1.111). User screen locking This Security Objective is fullfilled by the TOE.

7.4.24 Assurance of effective storage integrity

DP.Storage_Integrity (see 3.8.24). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Storage_Integrity (see 4.1.122). Storage integrity This Security Objective is fullfilled by the TOE.

7.4.25 System access banners

DP.Sys_Access_Banners (see 3.8.25). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Sys_Access_Banners (see 4.1.123). System access banners This Security Objective is fullfilled by the TOE.

7.4.26 Validation of security function integrity

DP.Sys_Assur_HW/SW/FW (see 3.8.26). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Sys_Assur_HW/SW/FW (see 4.1.124). Validation of security function This Security Objective is fullfilled by the TOE.

7.4.27 System backup procedures

DP.Sys_Backup_Procs (see 3.8.27). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Sys_Backup_Procs (see 4.1.125). System backup procedures This Security Objective is fullfilled by the TOE.

7.4.28 Restoration with minimal loss

DP.Sys_Backup_Restore (see 3.8.28). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Sys_Backup_Restore (see 4.1.126). Frequent backups to prevent minimal loss This Security Objective is fullfilled by the TOE.

7.4.29 Effective backup restoration

DP.Sys_Backup_Storage (see 3.8.29). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Sys_Backup_Storage (see 4.1.127). Sufficient backup storage and effective restoration This Security Objective is fullfilled by the TOE.

7.4.30 Protection from security function modification

DP.System_Protection (see 3.8.30). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Config_Management (see 4.1.48). Implement operational configuration management This Security Objective is fullfilled by the TOE.

O.Sys_Assur_HW/SW/FW (see 4.1.124). Validation of security function This Security Objective is fullfilled by the TOE.

O.Sys_Self_Protection (see 4.1.128). Protection of system security function This Security Objective is fullfilled by the TOE.

7.4.31 Trusted system recovery

DP.System_Recovery (see 3.8.31). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Trusted_Recovery_Doc (see 4.1.138). Documentation of untrusted data recovery This Security Objective is fullfilled by the TOE.

7.4.32 Encryption of transmitted user data

DP.User_Data_Dial-in (see 3.8.32). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.User_Data_Dial-in (see 4.1.142). Protection of user-session data This Security Objective is fullfilled by the TOE.

7.4.33 Protection of stored user data

DP.User_Data_Storage (see 3.8.33). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Info_Flow_Control (see 4.1.76). System enforced information flow This Security Objective is fullfilled by the TOE.

O.User_Data_Integrity (see 4.1.143). Integrity protection of stored user data This Security Objective is fullfilled by the TOE.

O.User_Defined_AC (see 4.1.145). User-defined access control This Security Objective is fullfilled by the TOE.

7.4.34 Protection of transmitted user data

DP.User_Data_Transfer (see 3.8.34). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Identify_Unusual_Act (see 4.1.75). Identify unusual user activity This Security Objective is fullfilled by the TOE.

O.User_Data_Transfer (see 4.1.144). Protection of transmitted user data This Security Objective is fullfilled by the TOE.

7.4.35 Discretionary access control

DP.User_Defined_AC (see 3.8.35). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.User_Defined_AC (see 4.1.145). User-defined access control This Security Objective is fullfilled by the TOE.

7.4.36 General user documentation

DP.User_Documentation (see 3.8.36). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.User_Guidance (see 4.1.146). User guidance documentation This Security Objective is fullfilled by the TOE.

7.4.37 Malicious code prevention

DP.Malicious_Code (see 3.10.1). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Malicious_Code (see 4.2.6). Procedures for preventing malicious code Compliance of this Security Objective is assigned to the Environment.

7.4.38 Backup protection and restoration

DP.Sys_Backup_Verify (see 3.10.2). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Sys_Backup_Verify (see 4.2.18). Detect modifications of backup hardware, firmware, software Compliance of this Security Objective is assigned to the Environment.

7.4.39 Enhanced user identification and authentication

DP.User_Auth_Enhanced (see 3.10.3). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.User_Auth_Enhanced (see 4.2.21). Enhanced user authentication Compliance of this Security Objective is assigned to the Environment.

7.4.40 Non-repudiation capabilities

DP.Non-Repudiation (see 3.10.4). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.NonRepudiate_Recd (see 4.2.10). Non-repudiation for received information Compliance of this Security Objective is assigned to the Environment.

O.NonRepudiate_Sent (see 4.1.101). Non-repudiation for sent information This Security Objective is fullfilled by the TOE.

7.4.41 Physical tampering detection and notification

DP.Tamper_ID (see 3.10.5). The following Security Objectives are considered sufficient to comply completely with this Detailed Policy Statement:

O.Tamper_ID (see 4.2.19). Tamper detection Compliance of this Security Objective is assigned to the Environment.

7.5 Security Objectives - Security Requirements Rationale

Up to this point, Security Objectives have been identified, to cover both technical Detailed Attacks against the TOE and Detailed Policy Statements. These Security Objectives have also been assigned both to the TOE and to the Environment, by application of the active Assumptions.

For those Security Objectives assigned to the TOE, Security Requirements already identified in Sec. 5 are selected to guarantee their satisfaction. This rationale includes coverage and completeness mappings.

7.5.1 Limitation of administrative access control

O.AC_Admin_Limit (see 4.1.1). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ACC.1 Subset access control This is actually satisfied by a higher component, (see 5.1.4).

FDP_ACF.1 Security attribute based access control (see 5.1.4).

7.5.2 Object security attributes and exportation

O.AC_Label_Export (see 4.1.2). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ACC.1 Subset access control This is actually satisfied by a higher component, (see 5.1.4).

FDP_ACF.1 Security attribute based access control (see 5.1.4).

FDP_ETC.2 Export of user data with security attributes (see 5.1.4).

7.5.3 Access history for user session

O.Access_History (see 4.1.3). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FTA_TAH.1 TOE access history (see 5.1.9).

7.5.4 Limit an administrator's ability to modify user-subject bindings

O.Adm_Limits_Bindings (see 4.1.4). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FIA_UAU.2 User authentication before any action (see 5.1.5).

FIA_UAU.6 Re-authenticating (see 5.1.5).

FIA_USB.1 User-subject binding (see 5.1.5).

FMT_MTD.1 Management of TSF data (see 5.1.6).

7.5.5 Limit administrator's modification of user attributes

O.Adm_User_Att_Mod (see 4.1.5). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FIA_ATD.1 User attribute definition (see 5.1.5).

FIA_UAU.2 User authentication before any action (see 5.1.5).

FMT_MSA.1 Management of security attributes (see 5.1.6).

7.5.6 Administrative validation of executables

O.Admin_Code_Val (see 4.1.6). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ACC.1 Subset access control This is actually satisfied by a higher component, (see 5.1.4).

FDP_ACF.1 Security attribute based access control (see 5.1.4).

FDP_SDI.1 Stored data integrity monitoring This is actually satisfied by a higher component, (see 5.1.4).

FMT_MSA.1 Management of security attributes (see 5.1.6).

FPT_TST.1 TSF testing (see 5.1.7).

7.5.7 Software validation for absence of steganography

O.Admin_Code_Val_Sten (see 4.1.7). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ACC.1 Subset access control This is actually satisfied by a higher component, (see 5.1.4).

FDP_ACF.1 Security attribute based access control (see 5.1.4).

FMT_MSA.1 Management of security attributes (see 5.1.6).

FMT_MSA.3 Static attribute initialisation (see 5.1.6).

FMT_SMR.1 Security roles This is actually satisfied by a higher component, (see 5.1.6).

7.5.8 Administrator guidance documentation

O.Admin_Guidance (see 4.1.8). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_ADM.1 Administrator guidance (see 5.2.4).

7.5.9 Apply patches to fix the code

O.Apply_Code_Fixes (see 4.1.9). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_ADM.1 Administrator guidance (see 5.2.4).

ALC_FLR.3 Systematic flaw remediation (see 5.2.5).

FMT_MOF.1 Management of security functions behaviour (see 5.1.6).

FMT_MSA.1 Management of security attributes (see 5.1.6).

7.5.10 CM automation

O.Assurance_ACM_AUT (see 4.1.10). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ACM_AUT.2 Complete CM automation (see 5.2.1).

7.5.11 CM capabilities

O.Assurance_ACM_CAP (see 4.1.11). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ACM_CAP.4 Generation support and acceptance procedures (see 5.2.1).

7.5.12 CM scope

O.Assurance_ACM_SCP (see 4.1.12). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ACM_SCP.3 Development tools CM coverage (see 5.2.1).

7.5.13 Delivery

O.Assurance_ADO_DEL (see 4.1.13). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADO_DEL.2 Detection of modification (see 5.2.2).

7.5.14 Installation, generation and start-up

O.Assurance_ADO_IGS (see 4.1.14). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADO_IGS.1 Installation, generation, and start-up procedures (see 5.2.2).

7.5.15 Functional specification

O.Assurance_ADV_FSP (see 4.1.15). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADV_FSP.3 Semiformal functional specification (see 5.2.3).

7.5.16 High-level design

O.Assurance_ADV_HLD (see 4.1.16). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADV_HLD.3 Semiformal high-level design (see 5.2.3).

7.5.17 Implementation representation

O.Assurance_ADV_IMP (see 4.1.17). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADV_IMP.3 Structured implementation of the TSF (see 5.2.3).

7.5.18 TSF internals

O.Assurance_ADV_INT (see 4.1.18). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADV_INT.3 Minimisation of complexity (see 5.2.3).

7.5.19 Low-level design

O.Assurance_ADV_LLD (see 4.1.19). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADV_LLD.1 Descriptive low-level design (see 5.2.3).

7.5.20 Representation correspondence

O.Assurance_ADV_RCR (see 4.1.20). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADV_RCR.2 Semiformal correspondence demonstration (see 5.2.3).

7.5.21 Security policy modeling

O.Assurance_ADV_SPM (see 4.1.21). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADV_SPM.1 Informal TOE security policy model This is actually satisfied by a higher component, (see 5.2.3).

7.5.22 Administrator guidance

O.Assurance_AGD_ADM (see 4.1.22). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_ADM.1 Administrator guidance (see 5.2.4).

7.5.23 User guidance

O.Assurance_AGD_USR (see 4.1.23). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_USR.1 User guidance (see 5.2.4).

7.5.24 Development security

O.Assurance_ALC_DVS (see 4.1.24). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ALC_DVS.2 Sufficiency of security measures (see 5.2.5).

7.5.25 Flaw remediation

O.Assurance_ALC_FLR (see 4.1.25). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ALC_FLR.3 Systematic flaw remediation (see 5.2.5).

7.5.26 Life cycle definition

O.Assurance_ALC_LCD (see 4.1.26). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ALC_LCD.3 Measurable life-cycle model (see 5.2.5).

7.5.27 Tools and techniques

O.Assurance_ALC_TAT (see 4.1.27). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ALC_TAT.2 Compliance with implementation standards (see 5.2.5).

7.5.28 Coverage

O.Assurance_ATE_COV (see 4.1.28). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ATE_COV.2 Analysis of coverage (see 5.2.6).

7.5.29 Depth

O.Assurance_ATE_DPT (see 4.1.29). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ATE_DPT.3 Testing: implementation representation (see 5.2.6).

7.5.30 Functional tests

O.Assurance_ATE_FUN (see 4.1.30). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ATE_FUN.2 Ordered functional testing (see 5.2.6).

7.5.31 Independent testing

O.Assurance_ATE_IND (see 4.1.31). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ATE_IND.2 Independent testing - sample (see 5.2.6).

7.5.32 Covert channel analysis

O.Assurance_AVA_CCA (see 4.1.32). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AVA_CCA.1 Covert channel analysis (see 5.2.7).

7.5.33 Misuse

O.Assurance_AVA_MSU (see 4.1.33). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AVA_MSU.2 Validation of analysis (see 5.2.7).

7.5.34 Strength of TOE security functions

O.Assurance_AVA_SOF (see 4.1.34). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AVA_SOF.1 Strength of TOE security function evaluation (see 5.2.7).

7.5.35 Vulnerability analysis

O.Assurance_AVA_VLA (see 4.1.35). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AVA_VLA.2 Independent vulnerability analysis (see 5.2.7).

7.5.36 Complete security functions or recover to previous state

O.Atomic_Functions (see 4.1.36). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_RCV.4 Function recovery (see 5.1.7).

7.5.37 Audit changes of system entry parameters

O.Aud_Sys_Entry_Parms (see 4.1.37). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FAU_GEN.1 Audit data generation (see 5.1.1).

FPT_STM.1 Reliable time stamps (see 5.1.7).

FMT_MTD.1 Management of TSF data (see 5.1.6).

FMT_MTD.2 Management of limits on TSF data (see 5.1.6).

FMT_MTD.3 Secure TSF data (see 5.1.6).

7.5.38 Auditing for user accountability

O.Audit_Account (see 4.1.38). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FAU_GEN.1 Audit data generation (see 5.1.1).

FPT_STM.1 Reliable time stamps (see 5.1.7).

FAU_GEN.2 User identity association (see 5.1.1).

FAU_SAR.1 Audit review (see 5.1.1).

FAU_SAR.3 Selectable audit review (see 5.1.1).

FAU_SEL.1 Selective audit (see 5.1.1).

FMT_MOF.1 Management of security functions behaviour (see 5.1.6).

FMT_SMR.1 Security roles This is actually satisfied by a higher component, (see 5.1.6).

7.5.39 Audit-administration role duties

O.Audit_Admin_Role (see 4.1.39). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_ADM.1 Administrator guidance (see 5.2.4).

FAU_STG.2 Guarantees of audit data availability (see 5.1.1).

FMT_MTD.1 Management of TSF data (see 5.1.6).

FMT_SMR.2 Restrictions on security roles (see 5.1.6).

7.5.40 Audit system access to deter misuse

O.Audit_Deter_Misuse (see 4.1.40). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FAU_GEN.2 User identity association (see 5.1.1).

FTA_TAB.1 Default TOE access banners (see 5.1.9).

7.5.41 Individual accountability

O.Audit_Gen_User (see 4.1.41). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FAU_GEN.2 User identity association (see 5.1.1).

FIA_UID.1 Timing of identification This is actually satisfied by a higher component, (see 5.1.5).

7.5.42 Audit records with identity

O.Audit_Generation (see 4.1.42). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FAU_GEN.1 Audit data generation (see 5.1.1).

FPT_STM.1 Reliable time stamps (see 5.1.7).

7.5.43 Respond to possible loss of stored audit records

O.Audit_Loss_Respond (see 4.1.43). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FAU_STG.4 Prevention of audit data loss (see 5.1.1).

7.5.44 Protect stored audit records

O.Audit_Protect (see 4.1.44). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FAU_STG.2 Guarantees of audit data availability (see 5.1.1).

7.5.45 User notification of data content changes

O.Change_Control_Users (see 4.1.45). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_DAU.2 Data authentication with identity of guarantor (see 5.1.4).

7.5.46 Code signing and verification

O.Code_Signing (see 4.1.46). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ETC.2 Export of user data with security attributes (see 5.1.4).

FDP_ITC.2 Import of user data with security attributes (see 5.1.4).

FDP_UIT.1 Data exchange integrity (see 5.1.4).

FIA_UAU.1 Timing of authentication This is actually satisfied by a higher component, (see 5.1.5).

7.5.47 Trusted channel to remote trusted system

O.Comm_Trusted_Channel (see 4.1.47). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FTP_ITC.1 Inter-TSF trusted channel (see 5.1.10).

7.5.48 Implement operational configuration management

O.Config_Management (see 4.1.48). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FMT_MOF.1 Management of security functions behaviour (see 5.1.6).

FMT_MTD.1 Management of TSF data (see 5.1.6).

7.5.49 Verify correct operation as designed

O.Correct_Operation (see 4.1.49). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_TST.1 TSF testing (see 5.1.7).

7.5.50 Cryptographic access control policy

O.Crypto_AC (see 4.1.50). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ACC.1 Subset access control This is actually satisfied by a higher component, (see 5.1.4).

FDP_ACF.1 Security attribute based access control (see 5.1.4).

7.5.51 Encrypted communications channel

O.Crypto_Comm_Channel (see 4.1.51). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FCS_CKM.1 Cryptographic key generation (see 5.1.3).

FCS_CKM.2 Cryptographic key distribution (see 5.1.3).

FCS_CKM.3 Cryptographic key access (see 5.1.3).

FCS_CKM.4 Cryptographic key destruction (see 5.1.3).

FCS_COP.1 Cryptographic operation (see 5.1.3).

7.5.52 Separation of cryptographic data

O.Crypto_Data_Sep (see 4.1.52). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADV_INT.1 Modularity This is actually satisfied by a higher component, (see 5.2.3).

FDP_IFF.3 Limited illicit information flows (see 5.1.4).

FPT_SEP.2 SFP domain separation This is actually satisfied by a higher component, (see 5.1.7).

7.5.53 Cryptographic Design and Implementation

O.Crypto_Dsgn_Impl (see 4.1.53). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADV_HLD.2 Security enforcing high-level design This is actually satisfied by a higher component, (see 5.2.3).

ADV_LLD.1 Descriptive low-level design (see 5.2.3).

ADV_RCR.1 Informal correspondence demonstration This is actually satisfied by a higher component, (see 5.2.3).

ALC_TAT.2 Compliance with implementation standards (see 5.2.5).

7.5.54 Cryptographic import, export, and inter-TSF transfer

O.Crypto_Import_Export (see 4.1.54). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_ADM.1 Administrator guidance (see 5.2.4).

AGD_USR.1 User guidance (see 5.2.4).

FDP_ETC.1 Export of user data without security attributes (see 5.1.4).

FDP_ETC.2 Export of user data with security attributes (see 5.1.4).

FDP_ITC.1 Import of user data without security attributes (see 5.1.4).

FDP_ITC.2 Import of user data with security attributes (see 5.1.4).

FTP_ITC.1 Inter-TSF trusted channel (see 5.1.10).

FTP_TRP.1 Trusted path (see 5.1.10).

7.5.55 Cryptographic Key Management

O.Crypto_Key_Man (see 4.1.55). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADV_FSP.1 Informal functional specification This is actually satisfied by a higher component, (see 5.2.3).

ADV_SPM.1 Informal TOE security policy model This is actually satisfied by a higher component, (see 5.2.3).

AVA_VLA.1 Developer vulnerability analysis This is actually satisfied by a higher component, (see 5.2.7).

FCS_CKM.1 Cryptographic key generation (see 5.1.3).

FCS_CKM.2 Cryptographic key distribution (see 5.1.3).

FCS_CKM.3 Cryptographic key access (see 5.1.3).

FCS_CKM.4 Cryptographic key destruction (see 5.1.3).

FDP_ACC.1 Subset access control This is actually satisfied by a higher component, (see 5.1.4).

FDP_ACF.1 Security attribute based access control (see 5.1.4).

FDP_ITC.1 Import of user data without security attributes (see 5.1.4).

FMT_MSA.1 Management of security attributes (see 5.1.6).

FMT_MTD.1 Management of TSF data (see 5.1.6).

FPT_SEP.1 TSF domain separation This is actually satisfied by a higher component, (see 5.1.7).

7.5.56 Management of cryptographic roles

O.Crypto_Manage_Roles (see 4.1.56). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FMT_SMR.1 Security roles This is actually satisfied by a higher component, (see 5.1.6).

7.5.57 Cryptographic Modular Design

O.Crypto_Modular_Dsgn (see 4.1.57). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADV_FSP.1 Informal functional specification This is actually satisfied by a higher component, (see 5.2.3).

ADV_HLD.1 Descriptive high-level design This is actually satisfied by a higher component, (see 5.2.3).

ADV_INT.1 Modularity This is actually satisfied by a higher component, (see 5.2.3).

ADV_LLD.1 Descriptive low-level design (see 5.2.3).

7.5.58 Cryptographic function definition

O.Crypto_Operation (see 4.1.58). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADV_FSP.1 Informal functional specification This is actually satisfied by a higher component, (see 5.2.3).

ADV_SPM.1 Informal TOE security policy model This is actually satisfied by a higher component, (see 5.2.3).

FCS_COP.1 Cryptographic operation (see 5.1.3).

7.5.59 Cryptographic self test

O.Crypto_Self_Test (see 4.1.59). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_SDI.1 Stored data integrity monitoring This is actually satisfied by a higher component, (see 5.1.4).

FPT_AMT.1 Abstract machine testing (see 5.1.7).

FPT_TST.1 TSF testing (see 5.1.7).

7.5.60 Test cryptographic functionality

O.Crypto_Test_Reqs (see 4.1.60). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ATE_COV.1 Evidence of coverage This is actually satisfied by a higher component, (see 5.2.6).

ATE_DPT.2 Testing: low-level design This is actually satisfied by a higher component, (see 5.2.6).

ATE_FUN.1 Functional testing This is actually satisfied by a higher component, (see 5.2.6).

ATE_IND.1 Independent testing - conformance This is actually satisfied by a higher component, (see 5.2.6).

AVA_VLA.1 Developer vulnerability analysis This is actually satisfied by a higher component, (see 5.2.7).

7.5.61 Enforce data exchange confidentiality

O.Data_Exchange_Conf (see 4.1.61). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FCS_COP.1 Cryptographic operation (see 5.1.3).

7.5.62 Control user data exportation

O.Data_Export_Control (see 4.1.62). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ETC.1 Export of user data without security attributes (see 5.1.4).

FDP_ETC.2 Export of user data with security attributes (see 5.1.4).

7.5.63 Data import/export to/from system control

O.Data_Imp_Exp_Control (see 4.1.63). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ETC.2 Export of user data with security attributes (see 5.1.4).

FDP_IFC.1 Subset information flow control (see 5.1.4).

FDP_IFF.1 Simple security attributes (see 5.1.4).

FDP_ITC.2 Import of user data with security attributes (see 5.1.4).

7.5.64 Sanitize data objects containing hidden or unused data

O.Export_Control (see 4.1.64). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ACC.1 Subset access control This is actually satisfied by a higher component, (see 5.1.4).

FDP_ETC.1 Export of user data without security attributes (see 5.1.4).

7.5.65 Label or mark information for external systems

O.External_Labels (see 4.1.65). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ETC.2 Export of user data with security attributes (see 5.1.4).

FDP_ITC.2 Import of user data with security attributes (see 5.1.4).

7.5.66 Preservation of secure state for failures in critical components

O.Fail_Secure (see 4.1.66). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_FLS.1 Failure with preservation of secure state (see 5.1.7).

7.5.67 Periodically check integrity

O.General_Integ_Checks (see 4.1.67). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_SDI.1 Stored data integrity monitoring This is actually satisfied by a higher component, (see 5.1.4).

FPT_TST.1 TSF testing (see 5.1.7).

7.5.68 Guarantee the availability of audit storage space

O.Guarantee_Audit_Stg (see 4.1.68). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FAU_STG.2 Guarantees of audit data availability (see 5.1.1).

FAU_STG.3 Action in case of possible audit data loss This is actually satisfied by a higher component, (see 5.1.1).

7.5.69 Limit sessions to outside users

O.Hack_Limit_Sessions (see 4.1.69). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_ADM.1 Administrator guidance (see 5.2.4).

FMT_MSA.1 Management of security attributes (see 5.1.6).

FTA_MCS.2 Per user attribute limitation on multiple concurrent sessions (see 5.1.9).

FTA_TSE.1 TOE session establishment (see 5.1.9).

7.5.70 Control hacker communication traffic

O.Hack_Traffic_Control (see 4.1.70). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_IFC.1 Subset information flow control (see 5.1.4).

FDP_IFF.1 Simple security attributes (see 5.1.4).

7.5.71 Identify and authenticate a user to support accountability

O.I&A_Domain (see 4.1.71). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FIA_ATD.1 User attribute definition (see 5.1.5).

FIA_AFL.1 Authentication failure handling (see 5.1.5).

FIA_SOS.1 Verification of secrets (see 5.1.5).

FIA_SOS.2 TSF Generation of secrets (see 5.1.5).

FIA_UAU.2 User authentication before any action (see 5.1.5).

FIA_UAU.7 Protected authentication feedback (see 5.1.5).

FIA_UID.1 Timing of identification This is actually satisfied by a higher component, (see 5.1.5).

FIA_USB.1 User-subject binding (see 5.1.5).

FTA_TAB.1 Default TOE access banners (see 5.1.9).

7.5.72 Transaction identification and authentication

O.I&A_Transaction (see 4.1.72). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_DAU.1 Basic data authentication This is actually satisfied by a higher component, (see 5.1.4).

7.5.73 Identify and authenticate each user

O.I&A_User (see 4.1.73). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FIA_UID.1 Timing of identification This is actually satisfied by a higher component, (see 5.1.5).

7.5.74 User-action identification and authentication

O.I&A_User_Action (see 4.1.74). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_ADM.1 Administrator guidance (see 5.2.4).

AGD_USR.1 User guidance (see 5.2.4).

FIA_UAU.2 User authentication before any action (see 5.1.5).

FIA_UID.2 User identification before any action (see 5.1.5).

FIA_USB.1 User-subject binding (see 5.1.5).

FMT_MOF.1 Management of security functions behaviour (see 5.1.6).

7.5.75 Identify unusual user activity

O.Identify_Unusual_Act (see 4.1.75). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FTA_TSE.1 TOE session establishment (see 5.1.9).

7.5.76 System enforced information flow

O.Info_Flow_Control (see 4.1.76). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_IFC.1 Subset information flow control (see 5.1.4).

FDP_IFF.1 Simple security attributes (see 5.1.4).

7.5.77 Provide information flow control administration

O.Info_Flow_Ctrl_Admin (see 4.1.77). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_IFC.1 Subset information flow control (see 5.1.4).

FDP_IFF.1 Simple security attributes (see 5.1.4).

FMT_MOF.1 Management of security functions behaviour (see 5.1.6).

FMT_SMR.1 Security roles This is actually satisfied by a higher component, (see 5.1.6).

7.5.78 Require inspection for absence of malicious code.

O.Input_Inspection (see 4.1.78). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ACC.1 Subset access control This is actually satisfied by a higher component, (see 5.1.4).

FDP_ACF.1 Security attribute based access control (see 5.1.4).

FDP_ITC.1 Import of user data without security attributes (see 5.1.4).

7.5.79 Data marking integrity export

O.Integ_Data_Mark_Exp (see 4.1.79). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ETC.2 Export of user data with security attributes (see 5.1.4).

7.5.80 Integrity of system data transferred externally

O.Integ_Sys_Data_Ext (see 4.1.80). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_ITI.1 Inter-TSF detection of modification (see 5.1.7).

7.5.81 Integrity of system data transferred internally

O.Integ_Sys_Data_Int (see 4.1.81). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_ITT.1 Basic internal TSF data transfer protection (see 5.1.7).

FPT_SSP.2 Mutual trusted acknowledgement (see 5.1.7).

7.5.82 Protect user data during internal transfer

O.Integ_User_Data_Int (see 4.1.82). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ITT.2 Transmission separation by attribute (see 5.1.4).

7.5.83 Correct attribute exchange with another trusted product

O.Integrity_Attr_Exch (see 4.1.83). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_TDC.1 Inter-TSF basic TSF data consistency (see 5.1.7).

7.5.84 Integrity protection for user data and software

O.Integrity_Data/SW (see 4.1.84). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_SDI.2 Stored data integrity monitoring and action (see 5.1.4).

7.5.85 Integrity of system data replication

O.Integrity_Data_Rep (see 4.1.85). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_TRC.1 Internal TSF consistency (see 5.1.7).

7.5.86 Operational integrity system function testing

O.Integrity_Practice (see 4.1.86). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_AMT.1 Abstract machine testing (see 5.1.7).

FPT_TST.1 TSF testing (see 5.1.7).

7.5.87 Isolate untrusted executables

O.Isolate_Executables (see 4.1.87). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ACC.1 Subset access control This is actually satisfied by a higher component, (see 5.1.4).

FDP_ACF.1 Security attribute based access control (see 5.1.4).

FMT_MSA.3 Static attribute initialisation (see 5.1.6).

FPT_SEP.1 TSF domain separation This is actually satisfied by a higher component, (see 5.1.7).

7.5.88 Lifecycle security

O.Lifecycle_Security (see 4.1.88). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ALC_DVS.1 Identification of security measures This is actually satisfied by a higher component, (see 5.2.5).

ALC_FLR.1 Basic flaw remediation This is actually satisfied by a higher component, (see 5.2.5).

ALC_LCD.1 Developer defined life-cycle model This is actually satisfied by a higher component, (see 5.2.5).

ALC_TAT.1 Well-defined development tools This is actually satisfied by a higher component, (see 5.2.5).

7.5.89 Restrict actions before authentication

O.Limit_Actions_Auth (see 4.1.89). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FIA_UAU.2 User authentication before any action (see 5.1.5).

7.5.90 Limit the number of user initiated communication sessions

O.Limit_Comm_Sessions (see 4.1.90). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FTA_MCS.2 Per user attribute limitation on multiple concurrent sessions (see 5.1.9).

FTA_TSE.1 TOE session establishment (see 5.1.9).

7.5.91 Limit multiple sessions

O.Limit_Mult_Sessions (see 4.1.91). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FTA_MCS.2 Per user attribute limitation on multiple concurrent sessions (see 5.1.9).

7.5.92 Maintain security domain

O.Maintain_Sec_Domain (see 4.1.92). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_SEP.3 Complete reference monitor (see 5.1.7).

7.5.93 Controlled access by maintenance personnel

O.Maintenance_Access (see 4.1.93). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FMT_MOF.1 Management of security functions behaviour (see 5.1.6).

FMT_SMR.1 Security roles This is actually satisfied by a higher component, (see 5.1.6).

7.5.94 Expiration of maintenance privileges

O.Maintenance_Recover (see 4.1.94). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FMT_SAE.1 Time-limited authorisation (see 5.1.6).

7.5.95 Manage resource security attributes

O.Manage_Res_Sec_Attr (see 4.1.95). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_USR.1 User guidance (see 5.2.4).

FAU_GEN.1 Audit data generation (see 5.1.1).

FPT_STM.1 Reliable time stamps (see 5.1.7).

FMT_MSA.1 Management of security attributes (see 5.1.6).

7.5.96 Manage security-critical data to avoid storage space being exceeded

O.Manage_TSF_Data (see 4.1.96). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FMT_MTD.2 Management of limits on TSF data (see 5.1.6).

7.5.97 Eliminate residual information

O.No_Residual_Info (see 4.1.97). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_RIP.2 Full residual information protection (see 5.1.4).

7.5.98 Non-repudiation support for sent information by the nonlocal receiving TSF.

O.NonRepud_Assess_Sent (see 4.1.98). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_ADM.1 Administrator guidance (see 5.2.4).

AGD_USR.1 User guidance (see 5.2.4).

FCO_NRO.1 Selective proof of origin This is actually satisfied by a higher component, (see 5.1.2).

FMT_MOF.1 Management of security functions behaviour (see 5.1.6).

7.5.99 Non-repudiation support for sent information by the sender's TSF.

O.NonRepud_Gen_Sent (see 4.1.99). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_ADM.1 Administrator guidance (see 5.2.4).

FCO_NRO.2 Enforced proof of origin (see 5.1.2).

FMT_MOF.1 Management of security functions behaviour (see 5.1.6).

7.5.100 Non-repudiation for sent information, local users

O.NonRepud_Locals_Sent (see 4.1.100). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_ADM.1 Administrator guidance (see 5.2.4).

AGD_USR.1 User guidance (see 5.2.4).

FCO_NRO.2 Enforced proof of origin (see 5.1.2).

FDP_ITT.1 Basic internal transfer protection This is actually satisfied by a higher component, (see 5.1.4).

FMT_MOF.1 Management of security functions behaviour (see 5.1.6).

7.5.101 Non-repudiation for sent information

O.NonRepudiate_Sent (see 4.1.101). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FCO_NRO.2 Enforced proof of origin (see 5.1.2).

7.5.102 Basic object attribute integrity

O.Obj_Attr_Integrity (see 4.1.102). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ACC.1 Subset access control This is actually satisfied by a higher component, (see 5.1.4).

FDP_ACF.1 Security attribute based access control (see 5.1.4).

FMT_MSA.1 Management of security attributes (see 5.1.6).

FMT_MSA.2 Secure security attributes (see 5.1.6).

FMT_MSA.3 Static attribute initialisation (see 5.1.6).

FMT_SMR.1 Security roles This is actually satisfied by a higher component, (see 5.1.6).

7.5.103 Object domain protection

O.Obj_Protection (see 4.1.103). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ACC.1 Subset access control This is actually satisfied by a higher component, (see 5.1.4).

FDP_ACF.1 Security attribute based access control (see 5.1.4).

FMT_MSA.3 Static attribute initialisation (see 5.1.6).

7.5.104 Provide priority of service

O.Priority_Of_Service (see 4.1.104). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FRU_PRS.2 Full priority of service (see 5.1.8).

7.5.105 Privileged-interface status

O.Prvlg_IF_Status (see 4.1.105). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FMT_MTD.1 Management of TSF data (see 5.1.6).

7.5.106 Identify message modification in messages received

O.Rcv_MsgMod_ID (see 4.1.106). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_UIT.1 Data exchange integrity (see 5.1.4).

7.5.107 Recovery from modification of received messages

O.Rcv_MsgMod_Rcvr (see 4.1.107). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_UIT.1 Data exchange integrity (see 5.1.4).

FDP_UIT.3 Destination data exchange recovery (see 5.1.4).

7.5.108 Provide reference monitor

O.Reference_Monitor (see 4.1.108). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_RVM.1 Non-bypassability of the TSP (see 5.1.7).

7.5.109 Disable remote execution

O.Remote_Execution (see 4.1.109). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ACC.1 Subset access control This is actually satisfied by a higher component, (see 5.1.4).

FDP_ACF.1 Security attribute based access control (see 5.1.4).

7.5.110 Resource quotas for users and services

O.Resource_Quotas (see 4.1.110). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FRU_RSA.2 Minimum and maximum quotas (see 5.1.8).

7.5.111 User screen locking

O.Screen_Lock (see 4.1.111). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FTA_SSL.1 TSF-initiated session locking (see 5.1.9).

FTA_SSL.2 User-initiated locking (see 5.1.9).

7.5.112 Security-relevant configuration management

O.Secure_Configuration (see 4.1.112). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_ADM.1 Administrator guidance (see 5.2.4).

FMT_MOF.1 Management of security functions behaviour (see 5.1.6).

FMT_MTD.1 Management of TSF data (see 5.1.6).

7.5.113 Protect and maintain secure system state

O.Secure_State (see 4.1.113). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_FLS.1 Failure with preservation of secure state (see 5.1.7).

FPT_RCV.1 Manual recovery (see 5.1.7).

FPT_RCV.4 Function recovery (see 5.1.7).

7.5.114 Manage security attributes

O.Security_Attr_Mgt (see 4.1.114). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FMT_MSA.1 Management of security attributes (see 5.1.6).

FMT_MSA.2 Secure security attributes (see 5.1.6).

FMT_MSA.3 Static attribute initialisation (see 5.1.6).

7.5.115 Manage security-critical data

O.Security_Data_Mgt (see 4.1.115). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FMT_MTD.1 Management of TSF data (see 5.1.6).

FMT_MTD.2 Management of limits on TSF data (see 5.1.6).

FMT_MTD.3 Secure TSF data (see 5.1.6).

7.5.116 Manage behavior of security functions

O.Security_Func_Mgt (see 4.1.116). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FMT_MOF.1 Management of security functions behaviour (see 5.1.6).

7.5.117 Security roles

O.Security_Roles (see 4.1.117). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FMT_SMR.2 Restrictions on security roles (see 5.1.6).

7.5.118 System terminates session for inactivity

O.Session_Termination (see 4.1.118). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FTA_SSL.3 TSF-initiated termination (see 5.1.9).

7.5.119 Identify message modification in messages sent

O.Snt_MsgMod_ID (see 4.1.119). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_UIT.1 Data exchange integrity (see 5.1.4).

7.5.120 Support recovery from modification of sent messages

O.Snt_MsgMod_Rcvr (see 4.1.120). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_UIT.1 Data exchange integrity (see 5.1.4).

FDP_UIT.3 Destination data exchange recovery (see 5.1.4).

7.5.121 Examine the source code for developer flaws

O.Source_Code_Exam (see 4.1.121). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ADV_IMP.1 Subset of the implementation of the TSF This is actually satisfied by a higher component, (see 5.2.3).

ADV_LLD.1 Descriptive low-level design (see 5.2.3).

ADV_RCR.1 Informal correspondence demonstration This is actually satisfied by a higher component, (see 5.2.3).

7.5.122 Storage integrity

O.Storage_Integrity (see 4.1.122). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_SDI.1 Stored data integrity monitoring This is actually satisfied by a higher component, (see 5.1.4).

7.5.123 System access banners

O.Sys_Access_Banners (see 4.1.123). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FTA_TAB.1 Default TOE access banners (see 5.1.9).

7.5.124 Validation of security function

O.Sys_Assur_HW/SW/FW (see 4.1.124). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

ATE_FUN.1 Functional testing This is actually satisfied by a higher component, (see 5.2.6).

FPT_TST.1 TSF testing (see 5.1.7).

7.5.125 System backup procedures

O.Sys_Backup_Procs (see 4.1.125). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_RCV.1 Manual recovery (see 5.1.7).

7.5.126 Frequent backups to prevent minimal loss

O.Sys_Backup_Restore (see 4.1.126). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FMT_MOF.1 Management of security functions behaviour (see 5.1.6).

FMT_MTD.1 Management of TSF data (see 5.1.6).

7.5.127 Sufficient backup storage and effective restoration

O.Sys_Backup_Storage (see 4.1.127). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FMT_MOF.1 Management of security functions behaviour (see 5.1.6).

FMT_MTD.1 Management of TSF data (see 5.1.6).

7.5.128 Protection of system security function

O.Sys_Self_Protection (see 4.1.128). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_SEP.2 SFP domain separation This is actually satisfied by a higher component, (see 5.1.7).

7.5.129 Limit administrator's modification of security-critical code or data

O.TSF_Mod_Limit (see 4.1.129). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FMT_MSA.2 Secure security attributes (see 5.1.6).

FMT_MTD.1 Management of TSF data (see 5.1.6).

FMT_MTD.2 Management of limits on TSF data (see 5.1.6).

FMT_MTD.3 Secure TSF data (see 5.1.6).

FMT_SMR.2 Restrictions on security roles (see 5.1.6).

7.5.130 Local detection of received security-critical data modified in transit

O.TSF_Rcv_Err_ID_Loc (see 4.1.130). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_ITI.1 Inter-TSF detection of modification (see 5.1.7).

7.5.131 Remote detection of received security-critical data modified in transit

O.TSF_Rcv_Err_ID_Rem (see 4.1.131). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_ITI.1 Inter-TSF detection of modification (see 5.1.7).

7.5.132 Local detection of sent security-critical data modified in transit

O.TSF_Snd_Err_ID_Loc (see 4.1.132). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_ITI.1 Inter-TSF detection of modification (see 5.1.7).

7.5.133 Remote detection of sent security-critical data modified in transit.

O.TSF_Snd_Err_ID_Rem (see 4.1.133). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_ITI.1 Inter-TSF detection of modification (see 5.1.7).

7.5.134 Trusted distributed system recovery

O.Trusted_DS_Recov (see 4.1.134). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_RCV.1 Manual recovery (see 5.1.7).

FPT_RCV.4 Function recovery (see 5.1.7).

7.5.135 Provide a trusted path

O.Trusted_Path (see 4.1.135). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FTP_TRP.1 Trusted path (see 5.1.10).

7.5.136 Trusted path and channel

O.Trusted_Path&Channel (see 4.1.136). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FTP_ITC.1 Inter-TSF trusted channel (see 5.1.10).

FTP_TRP.1 Trusted path (see 5.1.10).

7.5.137 Trusted recovery of security functionality

O.Trusted_Recovery (see 4.1.137). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FPT_RCV.1 Manual recovery (see 5.1.7).

7.5.138 Documentation of untrusted data recovery

O.Trusted_Recovery_Doc (see 4.1.138). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_ADM.1 Administrator guidance (see 5.2.4).

7.5.139 Maintain user attributes

O.User_Attributes (see 4.1.139). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FIA_ATD.1 User attribute definition (see 5.1.5).

FMT_MSA.1 Management of security attributes (see 5.1.6).

7.5.140 User authorization management

O.User_Auth_Management (see 4.1.140). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_ADM.1 Administrator guidance (see 5.2.4).

AGD_USR.1 User guidance (see 5.2.4).

FMT_MSA.1 Management of security attributes (see 5.1.6).

FMT_REV.1 Revocation (see 5.1.6).

FMT_SAE.1 Time-limited authorisation (see 5.1.6).

7.5.141 Basic confidentiality-breach prevention

O.User_Conf_Prevention (see 4.1.141). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ACC.1 Subset access control This is actually satisfied by a higher component, (see 5.1.4).

FDP_ACF.1 Security attribute based access control (see 5.1.4).

FDP_ETC.1 Export of user data without security attributes (see 5.1.4).

FDP_IFC.1 Subset information flow control (see 5.1.4).

FDP_IFF.1 Simple security attributes (see 5.1.4).

7.5.142 Protection of user-session data

O.User_Data_Dial-in (see 4.1.142). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_IFC.1 Subset information flow control (see 5.1.4).

FTA_TSE.1 TOE session establishment (see 5.1.9).

7.5.143 Integrity protection of stored user data

O.User_Data_Integrity (see 4.1.143). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_SDI.1 Stored data integrity monitoring This is actually satisfied by a higher component, (see 5.1.4).

7.5.144 Protection of transmitted user data

O.User_Data_Transfer (see 4.1.144). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ITT.1 Basic internal transfer protection This is actually satisfied by a higher component, (see 5.1.4).

7.5.145 User-defined access control

O.User_Defined_AC (see 4.1.145). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

FDP_ACC.2 Complete access control (see 5.1.4).

FDP_ACF.1 Security attribute based access control (see 5.1.4).

7.5.146 User guidance documentation

O.User_Guidance (see 4.1.146). To achieve this Security Objective, the following Security Requirements shall be implemented by the TOE:

AGD_USR.1 User guidance (see 5.2.4).

7.6 Security Requirements Dependency Analysis

Security Requirements, both of functional and assurance nature, have some implicit dependencies that must be preserved and followed if they are to be used at their full utility. This section presents an analysis of the selected Security Requirements, and their dependencies as defined in [CC].

7.6.1 ACM_AUT.2 - Complete CM automation

ACM_AUT.2 depends on:

- ACM_CAP.3 (satisfied by ACM_CAP.4 (see 5.2.1))

7.6.2 ACM_CAP.4 - Generation support and acceptance procedures

ACM_CAP.4 depends on:

- ACM_SCP.1 (satisfied by ACM_SCP.3 (see 5.2.1)) **AND**
- ALC_DVS.1 (satisfied by ALC_DVS.2 (see 5.2.5))

7.6.3 ACM_SCP.3 - Development tools CM coverage

ACM_SCP.3 depends on:

- ACM_CAP.3 (satisfied by ACM_CAP.4 (see 5.2.1))

7.6.4 ADO_DEL.2 - Detection of modification

ADO_DEL.2 depends on:

- ACM_CAP.3 (satisfied by ACM_CAP.4 (see 5.2.1))

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

ADO_IGS.1 depends on:

- AGD_ADM.1 (satisfied by AGD_ADM.1 (see 5.2.4))

7.6.6 ADV_FSP.3 - Semiformal functional specification

ADV_FSP.3 depends on:

- ADV_RCR.1 (satisfied by ADV_RCR.2 (see 5.2.3))

7.6.7 ADV_HLD.3 - Semiformal high-level design

ADV_HLD.3 depends on:

- ADV_FSP.3 (satisfied by ADV_FSP.3 (see 5.2.3)) **AND**
- ADV_RCR.2 (satisfied by ADV_RCR.2 (see 5.2.3))

7.6.8 ADV_IMP.3 - Structured implementation of the TSF

ADV_IMP.3 depends on:

- ADV_INT.1 (satisfied by ADV_INT.3 (see 5.2.3)) **AND**
- ADV_LLD.1 (satisfied by ADV_LLD.1 (see 5.2.3)) **AND**
- ADV_RCR.1 (satisfied by ADV_RCR.2 (see 5.2.3)) **AND**
- ALC_TAT.1 (satisfied by ALC_TAT.2 (see 5.2.5))

7.6.9 ADV_INT.3 - Minimisation of complexity

ADV_INT.3 depends on:

- ADV_IMP.2 (satisfied by ADV_IMP.3 (see 5.2.3)) **AND**
- ADV_LLD.1 (satisfied by ADV_LLD.1 (see 5.2.3))

7.6.10 ADV_LLD.1 - Descriptive low-level design

ADV_LLD.1 depends on:

- ADV_HLD.2 (satisfied by ADV_HLD.3 (see 5.2.3)) **AND**
- ADV_RCR.1 (satisfied by ADV_RCR.2 (see 5.2.3))

7.6.11 ADV_RCR.2 - Semiformal correspondence demonstration

No dependencies defined.

7.6.12 ADV_SPM.2 - Semiformal TOE security policy model

ADV_SPM.2 depends on:

- ADV_FSP.1 (satisfied by ADV_FSP.3 (see 5.2.3))

7.6.13 AGD_ADM.1 - Administrator guidance

AGD_ADM.1 depends on:

- ADV_FSP.1 (satisfied by ADV_FSP.3 (see 5.2.3))

7.6.14 AGD_USR.1 - User guidance

AGD_USR.1 depends on:

- ADV_FSP.1 (satisfied by ADV_FSP.3 (see 5.2.3))

7.6.15 ALC_DVS.2 - Sufficiency of security measures

No dependencies defined.

7.6.16 ALC_FLR.3 - Systematic flaw remediation

No dependencies defined.

7.6.17 ALC_LCD.3 - Measurable life-cycle model

No dependencies defined.

7.6.18 ALC_TAT.2 - Compliance with implementation standards

ALC_TAT.2 depends on:

- ADV_IMP.1 (satisfied by ADV_IMP.3 (see 5.2.3))

7.6.19 ATE_COV.2 - Analysis of coverage

ATE_COV.2 depends on:

- ADV_FSP.1 (satisfied by ADV_FSP.3 (see 5.2.3)) **AND**
- ATE_FUN.1 (satisfied by ATE_FUN.2 (see 5.2.6))

7.6.20 ATE_DPT.3 - Testing: implementation representation

ATE_DPT.3 depends on:

- ADV_HLD.2 (satisfied by ADV_HLD.3 (see 5.2.3)) **AND**
- ADV_IMP.2 (satisfied by ADV_IMP.3 (see 5.2.3)) **AND**
- ADV_LLD.1 (satisfied by ADV_LLD.1 (see 5.2.3)) **AND**
- ATE_FUN.1 (satisfied by ATE_FUN.2 (see 5.2.6))

7.6.21 ATE_FUN.2 - Ordered functional testing

No dependencies defined.

7.6.22 ATE_IND.2 - Independent testing - sample

ATE_IND.2 depends on:

- ADV_FSP.1 (satisfied by ADV_FSP.3 (see 5.2.3)) **AND**
- AGD_ADM.1 (satisfied by AGD_ADM.1 (see 5.2.4)) **AND**
- AGD_USR.1 (satisfied by AGD_USR.1 (see 5.2.4)) **AND**
- ATE_FUN.1 (satisfied by ATE_FUN.2 (see 5.2.6))

7.6.23 AVA_CCA.1 - Covert channel analysis

AVA_CCA.1 depends on:

- ADV_FSP.2 (satisfied by ADV_FSP.3 (see 5.2.3)) **AND**
- ADV_IMP.2 (satisfied by ADV_IMP.3 (see 5.2.3)) **AND**
- AGD_ADM.1 (satisfied by AGD_ADM.1 (see 5.2.4)) **AND**
- AGD_USR.1 (satisfied by AGD_USR.1 (see 5.2.4))

7.6.24 AVA_MSU.2 - Validation of analysis

AVA_MSU.2 depends on:

- ADO_IGS.1 (satisfied by ADO_IGS.1 (see 5.2.2)) **AND**
- ADV_FSP.1 (satisfied by ADV_FSP.3 (see 5.2.3)) **AND**
- AGD_ADM.1 (satisfied by AGD_ADM.1 (see 5.2.4)) **AND**
- AGD_USR.1 (satisfied by AGD_USR.1 (see 5.2.4))

7.6.25 AVA_SOF.1 - Strength of TOE security function evaluation

AVA_SOF.1 depends on:

- ADV_FSP.1 (satisfied by ADV_FSP.3 (see 5.2.3)) **AND**
- ADV_HLD.1 (satisfied by ADV_HLD.3 (see 5.2.3))

7.6.26 AVA_VLA.2 - Independent vulnerability analysis

AVA_VLA.2 depends on:

- ADV_FSP.1 (satisfied by ADV_FSP.3 (see 5.2.3)) **AND**
- ADV_HLD.2 (satisfied by ADV_HLD.3 (see 5.2.3)) **AND**
- ADV_IMP.1 (satisfied by ADV_IMP.3 (see 5.2.3)) **AND**
- ADV_LLD.1 (satisfied by ADV_LLD.1 (see 5.2.3)) **AND**
- AGD_ADM.1 (satisfied by AGD_ADM.1 (see 5.2.4)) **AND**
- AGD_USR.1 (satisfied by AGD_USR.1 (see 5.2.4))

7.6.27 FAU_GEN.1 - Audit data generation

FAU_GEN.1 depends on:

- FPT_STM.1 (satisfied by FPT_STM.1 (see 5.1.7))

7.6.28 FAU_GEN.2 - User identity association

FAU_GEN.2 depends on:

- FAU_GEN.1 (satisfied by FAU_GEN.1 (see 5.1.1)) **AND**
- FIA_UID.1 (satisfied by FIA_UID.2 (see 5.1.5))

7.6.29 FAU_SAR.1 - Audit review

FAU_SAR.1 depends on:

- FAU_GEN.1 (satisfied by FAU_GEN.1 (see 5.1.1))

7.6.30 FAU_SAR.3 - Selectable audit review

FAU_SAR.3 depends on:

- FAU_SAR.1 (satisfied by FAU_SAR.1 (see 5.1.1))

7.6.31 FAU_SEL.1 - Selective audit

FAU_SEL.1 depends on:

- FAU_GEN.1 (satisfied by FAU_GEN.1 (see 5.1.1)) **AND**
- FMT_MTD.1 (satisfied by FMT_MTD.1 (see 5.1.6))

7.6.32 FAU_STG.2 - Guarantees of audit data availability

FAU_STG.2 depends on:

- FAU_GEN.1 (satisfied by FAU_GEN.1 (see 5.1.1))

7.6.33 FAU_STG.4 - Prevention of audit data loss

FAU_STG.4 depends on:

- FAU_STG.1 (satisfied by FAU_STG.2 (see 5.1.1))

7.6.34 FCO_NRO.2 - Enforced proof of origin

FCO_NRO.2 depends on:

- FIA_UID.1 (satisfied by FIA_UID.2 (see 5.1.5))

7.6.35 FCS_CKM.1 - Cryptographic key generation

FCS_CKM.1 depends on:

- FCS_CKM.2 (satisfied by FCS_CKM.2 (see 5.1.3)) **OR** FCS_COP.1 (satisfied by FCS_COP.1 (see 5.1.3)) **AND**
- FCS_CKM.4 (satisfied by FCS_CKM.4 (see 5.1.3)) **AND**
- FMT_MSA.2 (satisfied by FMT_MSA.2 (see 5.1.6))

7.6.36 FCS_CKM.2 - Cryptographic key distribution

FCS_CKM.2 depends on:

- FDP_ITC.1 (satisfied by FDP_ITC.1 (see 5.1.4)) **OR** FCS_CKM.1 (satisfied by FCS_CKM.1 (see 5.1.3)) **AND**
- FCS_CKM.4 (satisfied by FCS_CKM.4 (see 5.1.3)) **AND**
- FMT_MSA.2 (satisfied by FMT_MSA.2 (see 5.1.6))

7.6.37 FCS_CKM.3 - Cryptographic key access

FCS_CKM.3 depends on:

- FDP_ITC.1 (satisfied by FDP_ITC.1 (see 5.1.4)) **OR** FCS_CKM.1 (satisfied by FCS_CKM.1 (see 5.1.3)) **AND**
- FCS_CKM.4 (satisfied by FCS_CKM.4 (see 5.1.3)) **AND**
- FMT_MSA.2 (satisfied by FMT_MSA.2 (see 5.1.6))

7.6.38 FCS_CKM.4 - Cryptographic key destruction

FCS_CKM.4 depends on:

- FDP_ITC.1 (satisfied by FDP_ITC.1 (see 5.1.4)) **OR** FCS_CKM.1 (satisfied by FCS_CKM.1 (see 5.1.3)) **AND**
- FMT_MSA.2 (satisfied by FMT_MSA.2 (see 5.1.6))

7.6.39 FCS_COP.1 - Cryptographic operation

FCS_COP.1 depends on:

- FDP_ITC.1 (satisfied by FDP_ITC.1 (see 5.1.4)) **OR** FCS_CKM.1 (satisfied by FCS_CKM.1 (see 5.1.3)) **AND**
- FCS_CKM.4 (satisfied by FCS_CKM.4 (see 5.1.3)) **AND**
- FMT_MSA.2 (satisfied by FMT_MSA.2 (see 5.1.6))

7.6.40 FDP_ACC.2 - Complete access control

FDP_ACC.2 depends on:

- FDP_ACF.1 (satisfied by FDP_ACF.1 (see 5.1.4))

7.6.41 FDP_ACF.1 - Security attribute based access control

FDP_ACF.1 depends on:

- FDP_ACC.1 (satisfied by FDP_ACC.2 (see 5.1.4)) **AND**
- FMT_MSA.3 (satisfied by FMT_MSA.3 (see 5.1.6))

7.6.42 FDP_DAU.2 - Data authentication with identity of guarantor

FDP_DAU.2 depends on:

- FIA_UID.1 (satisfied by FIA_UID.2 (see 5.1.5))

7.6.43 FDP_ETC.1 - Export of user data without security attributes

FDP_ETC.1 depends on:

- FDP_ACC.1 (satisfied by FDP_ACC.2 (see 5.1.4)) **OR** FDP_IFC.1 (satisfied by FDP_IFC.1 (see 5.1.4))

7.6.44 FDP_ETC.2 - Export of user data with security attributes

FDP_ETC.2 depends on:

- FDP_ACC.1 (satisfied by FDP_ACC.2 (see 5.1.4)) **OR** FDP_IFC.1 (satisfied by FDP_IFC.1 (see 5.1.4))

7.6.45 FDP_IFC.1 - Subset information flow control

FDP_IFC.1 depends on:

- FDP_IFF.1 (satisfied by FDP_IFF.1 (see 5.1.4))

7.6.46 FDP_IFF.1 - Simple security attributes

FDP_IFF.1 depends on:

- FDP_IFC.1 (satisfied by FDP_IFC.1 (see 5.1.4)) **AND**
- FMT_MSA.3 (satisfied by FMT_MSA.3 (see 5.1.6))

7.6.47 FDP_IFF.3 - Limited illicit information flows

FDP_IFF.3 depends on:

- AVA_CCA.1 (satisfied by AVA_CCA.1 (see 5.2.7)) **AND**
- FDP_IFC.1 (satisfied by FDP_IFC.1 (see 5.1.4))

7.6.48 FDP_ITC.1 - Import of user data without security attributes

FDP_ITC.1 depends on:

- FDP_ACC.1 (satisfied by FDP_ACC.2 (see 5.1.4)) **OR** FDP_IFC.1 (satisfied by FDP_IFC.1 (see 5.1.4)) **AND**
- FMT_MSA.3 (satisfied by FMT_MSA.3 (see 5.1.6))

7.6.49 FDP_ITC.2 - Import of user data with security attributes

FDP_ITC.2 depends on:

- FDP_ACC.1 (satisfied by FDP_ACC.2 (see 5.1.4)) **OR** FDP_IFC.1 (satisfied by FDP_IFC.1 (see 5.1.4)) **AND**
- FTP_ITC.1 (satisfied by FTP_ITC.1 (see 5.1.10)) **OR** FTP_TRP.1 (satisfied by FTP_TRP.1 (see 5.1.10)) **AND**
- FPT_TDC.1 (satisfied by FPT_TDC.1 (see 5.1.7))

7.6.50 FDP_ITT.2 - Transmission separation by attribute

FDP_ITT.2 depends on:

- FDP_ACC.1 (satisfied by FDP_ACC.2 (see 5.1.4)) **OR** FDP_IFC.1 (satisfied by FDP_IFC.1 (see 5.1.4))

7.6.51 FDP_RIP.2 - Full residual information protection

No dependencies defined.

7.6.52 FDP_SDI.2 - Stored data integrity monitoring and action

No dependencies defined.

7.6.53 FDP_UIT.1 - Data exchange integrity

FDP_UIT.1 depends on:

- FDP_ACC.1 (satisfied by FDP_ACC.2 (see 5.1.4)) **OR** FDP_IFC.1 (satisfied by FDP_IFC.1 (see 5.1.4)) **AND**
- FTP_ITC.1 (satisfied by FTP_ITC.1 (see 5.1.10)) **OR** FTP_TRP.1 (satisfied by FTP_TRP.1 (see 5.1.10))

7.6.54 FDP_UIT.3 - Destination data exchange recovery

FDP_UIT.3 depends on:

- FDP_ACC.1 (satisfied by FDP_ACC.2 (see 5.1.4)) **OR** FDP_IFC.1 (satisfied by FDP_IFC.1 (see 5.1.4)) **AND**
- FDP_UIT.1 (satisfied by FDP_UIT.1 (see 5.1.4)) **AND**
- FTP_ITC.1 (satisfied by FTP_ITC.1 (see 5.1.10))

7.6.55 FIA_AFL.1 - Authentication failure handling

FIA_AFL.1 depends on:

- FIA_UAU.1 (satisfied by FIA_UAU.2 (see 5.1.5))

7.6.56 FIA_ATD.1 - User attribute definition

No dependencies defined.

7.6.57 FIA_SOS.1 - Verification of secrets

No dependencies defined.

7.6.58 FIA_SOS.2 - TSF Generation of secrets

No dependencies defined.

7.6.59 FIA_UAU.2 - User authentication before any action

FIA_UAU.2 depends on:

- FIA_UID.1 (satisfied by FIA_UID.2 (see 5.1.5))

7.6.60 FIA_UAU.6 - Re-authenticating

No dependencies defined.

7.6.61 FIA_UAU.7 - Protected authentication feedback

FIA_UAU.7 depends on:

- FIA_UID.1 (satisfied by FIA_UID.2 (see 5.1.5))

7.6.62 FIA_UID.2 - User identification before any action

No dependencies defined.

7.6.63 FIA_USB.1 - User-subject binding

FIA_USB.1 depends on:

- FIA_ATD.1 (satisfied by FIA_ATD.1 (see 5.1.5))

7.6.64 FMT_MOF.1 - Management of security functions behaviour

FMT_MOF.1 depends on:

- FMT_SMR.1 (satisfied by FMT_SMR.2 (see 5.1.6))

7.6.65 FMT_MSA.1 - Management of security attributes

FMT_MSA.1 depends on:

- FDP_ACC.1 (satisfied by FDP_ACC.2 (see 5.1.4)) **OR** FDP_IFC.1 (satisfied by FDP_IFC.1 (see 5.1.4)) **AND**
- FMT_SMR.1 (satisfied by FMT_SMR.2 (see 5.1.6))

7.6.66 FMT_MSA.2 - Secure security attributes

FMT_MSA.2 depends on:

- ADV_SPM.1 (satisfied by ADV_SPM.2 (see 5.2.3)) **AND**
- FDP_ACC.1 (satisfied by FDP_ACC.2 (see 5.1.4)) **OR** FDP_IFC.1 (satisfied by FDP_IFC.1 (see 5.1.4)) **AND**
- FMT_MSA.1 (satisfied by FMT_MSA.1 (see 5.1.6)) **AND**
- FMT_SMR.1 (satisfied by FMT_SMR.2 (see 5.1.6))

7.6.67 FMT_MSA.3 - Static attribute initialisation

FMT_MSA.3 depends on:

- FMT_MSA.1 (satisfied by FMT_MSA.1 (see 5.1.6)) **AND**
- FMT_SMR.1 (satisfied by FMT_SMR.2 (see 5.1.6))

7.6.68 FMT_MTD.1 - Management of TSF data

FMT_MTD.1 depends on:

- FMT_SMR.1 (satisfied by FMT_SMR.2 (see 5.1.6))

7.6.69 FMT_MTD.2 - Management of limits on TSF data

FMT_MTD.2 depends on:

- FMT_MTD.1 (satisfied by FMT_MTD.1 (see 5.1.6)) **AND**
- FMT_SMR.1 (satisfied by FMT_SMR.2 (see 5.1.6))

7.6.70 FMT_MTD.3 - Secure TSF data

FMT_MTD.3 depends on:

- ADV_SPM.1 (satisfied by ADV_SPM.2 (see 5.2.3)) **AND**
- FMT_MTD.1 (satisfied by FMT_MTD.1 (see 5.1.6))

7.6.71 FMT_REV.1 - Revocation

FMT_REV.1 depends on:

- FMT_SMR.1 (satisfied by FMT_SMR.2 (see 5.1.6))

7.6.72 FMT_SAE.1 - Time-limited authorisation

FMT_SAE.1 depends on:

- FMT_SMR.1 (satisfied by FMT_SMR.2 (see 5.1.6)) **AND**
- FPT_STM.1 (satisfied by FPT_STM.1 (see 5.1.7))

7.6.73 FMT_SMR.2 - Restrictions on security roles

FMT_SMR.2 depends on:

- FIA_UID.1 (satisfied by FIA_UID.2 (see 5.1.5))

7.6.74 FPT_AMT.1 - Abstract machine testing

No dependencies defined.

7.6.75 FPT_FLS.1 - Failure with preservation of secure state

FPT_FLS.1 depends on:

- ADV_SPM.1 (satisfied by ADV_SPM.2 (see 5.2.3))

7.6.76 FPT_ITI.1 - Inter-TSF detection of modification

No dependencies defined.

7.6.77 FPT_ITT.1 - Basic internal TSF data transfer protection

No dependencies defined.

7.6.78 FPT_RCV.1 - Manual recovery

FPT_RCV.1 depends on:

- FPT_TST.1 (satisfied by FPT_TST.1 (see 5.1.7)) **AND**
- AGD_ADM.1 (satisfied by AGD_ADM.1 (see 5.2.4)) **AND**
- ADV_SPM.1 (satisfied by ADV_SPM.2 (see 5.2.3))

7.6.79 FPT_RCV.4 - Function recovery

FPT_RCV.4 depends on:

- ADV_SPM.1 (satisfied by ADV_SPM.2 (see 5.2.3))

7.6.80 FPT_RVM.1 - Non-bypassability of the TSP

No dependencies defined.

7.6.81 FPT_SEP.3 - Complete reference monitor

No dependencies defined.

7.6.82 FPT_SSP.2 - Mutual trusted acknowledgement

FPT_SSP.2 depends on:

- FPT_ITT.1 (satisfied by FPT_ITT.1 (see 5.1.7))

7.6.83 FPT_STM.1 - Reliable time stamps

No dependencies defined.

7.6.84 FPT_TDC.1 - Inter-TSF basic TSF data consistency

No dependencies defined.

7.6.85 FPT_TRC.1 - Internal TSF consistency

FPT_TRC.1 depends on:

- FPT_ITT.1 (satisfied by FPT_ITT.1 (see 5.1.7))

7.6.86 FPT_TST.1 - TSF testing

FPT_TST.1 depends on:

- FPT_AMT.1 (satisfied by FPT_AMT.1 (see 5.1.7))

7.6.87 FRU_PRS.2 - Full priority of service

No dependencies defined.

7.6.88 FRU_RSA.2 - Minimum and maximum quotas

No dependencies defined.

7.6.89 FTA_MCS.2 - Per user attribute limitation on multiple concurrent sessions

FTA_MCS.2 depends on:

- FIA_UID.1 (satisfied by FIA_UID.2 (see 5.1.5))

7.6.90 FTA_SSL.1 - TSF-initiated session locking

FTA_SSL.1 depends on:

- FIA_UAU.1 (satisfied by FIA_UAU.2 (see 5.1.5))

7.6.91 FTA_SSL.2 - User-initiated locking

FTA_SSL.2 depends on:

- FIA_UAU.1 (satisfied by FIA_UAU.2 (see 5.1.5))

7.6.92 FTA_SSL.3 - TSF-initiated termination

No dependencies defined.

7.6.93 FTA_TAB.1 - Default TOE access banners

No dependencies defined.

7.6.94 FTA_TAH.1 - TOE access history

No dependencies defined.

7.6.95 FTA_TSE.1 - TOE session establishment

No dependencies defined.

7.6.96 FTP_ITC.1 - Inter-TSF trusted channel

No dependencies defined.

7.6.97 FTP_TRP.1 - Trusted path

No dependencies defined.

Index