Groove Cryptographic Services
(GrooveMisc.dll 2.5.0.1774; cryptopp.dll 5.0.4.0)
Common Criteria
Security Target
Version 3.4

Jean E. Petty

August 22, 2003

CYGNACOM
SOLUTIONS
an Entrust company

TABLE OF CONTENTS

SECTION **PAGE**

1 SECURITY TARGET INTRODUCTION 1

1.1 SECURITY TARGET IDENTIFICATION 1
1.2 SECURITY TARGET OVERVIEW 1
1.3 COMMON CRITERIA CONFORMANCE 1
1.4 RELATED DOCUMENTS 2

2 TOE DESCRIPTION 3

2.1 PRODUCT TYPE 3
2.2 GENERAL TOE FUNCTIONALITY 3
2.2.1 TSF, TSF Subsystems, and TSFI 4
2.2.2 GrooveMisc.dll (TSF-Subset) and its COM API 4
2.2.3 cryptopp.dll and its C++ API 5
2.2.4 Relationship Between the Two TSF Subsystems 5
2.3 CRYPTOGRAPHIC OPERATIONS 6
2.4 ENVIRONMENT SECURITY FUNCTIONAL REQUIREMENTS 7

3 SECURITY ENVIRONMENT 9

3.1 THREATS TO SECURITY 9
3.2 ASSUMPTIONS FOR THE IT ENVIRONMENT 9
3.3 THREATS TO THE SECURITY ENVIRONMENT 10

4 SECURITY OBJECTIVES 11

4.1 SECURITY OBJECTIVES FOR THE TOE 11
4.2 SECURITY OBJECTIVES FOR THE ENVIRONMENT 12

5 IT SECURITY REQUIREMENTS 13

5.1 TOE SECURITY FUNCTIONAL REQUIREMENTS 13
5.1.1 FCS – Cryptographic Support 13
5.1.2 FPT – Protection of the TOE Security Functions 17
5.2 SECURITY FUNCTIONAL REQUIREMENTS FOR THE ENVIRONMENT 18
5.2.1 FCS – Cryptographic Support 18
5.2.2 FDP – User Data Protection 18
5.2.3 FIA – Identification and Authentication 20
5.2.4 FMT – Security Management 20
5.3 STRENGTH OF FUNCTION REQUIREMENT 21
5.4 TOE SECURITY ASSURANCE REQUIREMENTS 22

6 TOE SUMMARY SPECIFICATION 23

6.1 TOE IT SECURITY FUNCTIONS 23
6.1.1 Key Generation 23
6.1.2 Cryptographic Operations 24
6.1.3 Self-Tests 25
6.2 ENVIRONMENTAL SECURITY FUNCTIONS 26
6.2.1 Cryptographic Key Destruction 26
6.2.2 Access Control 26
6.2.3 Data and Key Export 27
6.2.4 Data and Key Import 27
6.2.5 Identification and Authentication 27

6.2.6 Security Management *28*
6.3 STRENGTH OF FUNCTION REQUIREMENT 28
6.4 ASSURANCE MEASURES 28

7 PP CLAIMS **30**

8 RATIONALE **31**

8.1 SECURITY OBJECTIVES RATIONALE 31
8.1.1 All Assumptions and Threats Addressed *31*
8.1.2 All Objectives Necessary *34*
8.2 SECURITY REQUIREMENTS RATIONALE 36
8.2.1 All Objectives Met by Security Requirements *36*
8.2.2 All Functional Components Necessary *38*
8.2.3 Satisfaction of Dependencies *40*
8.2.4 Assurance Rationale *42*
8.3 TOE SUMMARY SPECIFICATION RATIONALE 42
8.3.1 All TOE Security Functional Requirements Satisfied *42*
8.3.2 All Security Functional Requirements for the Environment Satisfied *43*
8.3.3 All TOE and Environment Summary Specification (TSS) Functions Necessary *45*
8.3.4 Strength of Function Rationale *46*
8.3.5 Rationale for Explicitly Stated Requirement *46*
8.3.6 Assurance Measures Rationale *47*
8.4 PP CLAIMS RATIONALE 48

9 ACRONYMS **49**

10 REFERENCES **50**

TABLE OF FIGURES AND TABLES

TABLE/FIGURE **PAGE**

INTERPRETATIONS VI
NIAP INTERPRETATIONS VI
INTERNATIONAL INTERPRETATIONS VII
REVISION HISTORY VIII
FIGURE 1. TOE/TSF DIAGRAM 3
TABLE 3.1 – THREATS TO SECURITY 9
TABLE 3.2 –ASSUMPTIONS FOR THE IT ENVIRONMENT 9
TABLE 3.3 –THREATS TO THE IT SECURITY ENVIRONMENT 10
TABLE 4.1 – SECURITY OBJECTIVES FOR THE TOE 11
TABLE 4.2 – SECURITY OBJECTIVES FOR THE ENVIRONMENT 12
TABLE 5.1 – TOE SECURITY FUNCTIONAL REQUIREMENTS 13
TABLE 5.2 – SECURITY FUNCTIONAL REQUIREMENTS FOR THE ENVIRONMENT 18
TABLE 5.3 - ASSURANCE COMPONENTS 22
TABLE 6.1 – ASSURANCE EVALUATION EVIDENCE 29
TABLE 8.1 – ALL ASSUMPTIONS AND THREATS TO SECURITY COUNTERED BY OBJECTIVES 32
TABLE 8.2 – ALL IT SECURITY OBJECTIVES NECESSARY 34
TABLE 8.3– MAPPING OF IT SECURITY OBJECTIVES TO REQUIREMENTS 36
TABLE 8.4– MAPPING OF FUNCTIONAL REQUIREMENTS TO IT SECURITY OBJECTIVES 39
TABLE 8.5 – FUNCTIONAL REQUIREMENTS DEPENDENCIES 40
TABLE 8.6 – MAPPING OF FUNCTIONAL REQUIREMENTS TO TOE SUMMARY SPECIFICATION 42
TABLE 8.7 – MAPPING OF FUNCTIONAL REQUIREMENTS TO IT ENVIRONMENT SUMMARY SPECIFICATION 44
TABLE 8.8 – MAPPING OF TOE AND ENVIRONMENT SUMMARY SPECIFICATION TO FUNCTIONAL REQUIREMENTS 46
TABLE 8.9 – ASSURANCE EVALUATION EVIDENCE 47

Interpretations

This ST conforms with the NIAP and International interpretations listed in the following two tables.

NIAP Interpretations

#	Title
I-0347	Including Sensitive Information In Audit Records
I-0350	Clarification Of Resources/Objects For Residual Information Protection
I-0352	Rules Governing Binding Should Be Specifiable
I-0375	Elements Requiring Authentication Mechanism
I-0381	Relationship Between FPT_PHP And FMT_MOF
I-0389	Recovery To A Known State
I-0393	A Completely Evaluated ST Is Not Required When TOE Evaluation Starts
I-0395	Security Attributes Include Attributes Of Information And Resources
I-0405	American English Is An Acceptable Refinement
I-0406	Automated Or Manual Recovery Is Acceptable
I-0407	Empty Selections Or Assignments
I-0409	Other Properties In FMT_MSA.3 Should Be Specified By Assignment
I-0410	Auditing Of Subject Identity For Unsuccessful Logins
I-0411	Guidance Includes AGD_ADM, AGD_USR, ADO, And ALC_FLR
I-0412	Configuration Items In The Absence Of Configuration Management
I-0414	Site-Configurable Prevention Of Audit Loss
I-0415	User Attributes To Be Bound Should Be Specified
I-0416	Association Of Access Control Attributes With Subjects And Objects
I-0417	**Association Of Information Flow Attributes W/Subjects And Information**
I- 0418	Evaluation Of The TOE Summary Specification: Part 1 Vs Part 3
I-0420	Attribute Inheritance/Modification Rules Need To Be Included In Policy
I-0421	Application Notes In Protection Profiles Are Informative Only
I-0422	Clarification Of ``Audit Records''
I-0423	Some Modifications To The Audit Trail Are Authorized
I-0424	FPT_SEP.2 And FPT_SEP.3 Are Not Hierarchical
I-0425	Settable Failure Limits Are Permitted
I-0426	Content Of PP Claims Rationale
I-0427	Identification Of Standards
I-0429	Selecting One Or More
I-0459	CM Systems May Have Varying Degrees Of Rigor And Function

International Interpretations

#	Title
003	Unique identification of configuration items in the configuration list
004	ACM_SCP.*.1C requirements unclear
006	Virtual machine description
008	Augmented and Conformant overlap
009	Definition of Counter
013	Multiple SOF claims for multiple domains in a single TOE
016	Objective for ADO_DEL
019	Assurance Iterations
024	COTS product in TOE providing security
025	Level of detail required for hardware descriptions
027	Events and actions
031	Obvious vulnerabilities
032	Strength of Function Analysis in ASE_TSS
033	CC use of "Check"
037	ACM on Product or TOE?
043	Meaning of "clearly stated" in APE/ASE_OBJ.1
049	Threats met by environment
051	Use of documentation without C & P elements.
055	Incorrect Component referenced in Part 2 Annexes, FPT_RCV
058	Confusion over refinement
064	Apparent higher standard for explicitly stated requirements
065	No component to call out security function management
067	Application notes missing
069	Informal Security Policy Model
074	Duplicate informative text for ATE_COV.2-3 and ATE_DPT.1-3
075	Duplicate informative text for different work units
084	Aspects of objectives in TOE and environment
085	SOF Claims additional to the overall claim
095	SCP Dependency in ACM_CAP
098	Limitation of refinement
116	Indistinguishable work units for ADO_DEL
120	Sampling of process expectations unclear
127	Work unit not at the right place
128	Coverage of the delivery procedures
133	Consistency analysis in AVA_MSU.2
138	Iteration and narrowing of scope

Revision History

Date	Version	Description
December 6, 2001	0.1	Groove Snapshot Draft Document
December 14, 2001	0.2	Groove Initial Draft
January 4 2002	0.3	Groove Pre-Release Draft
January 9, 2002	1.0	Groove Release Draft
April 22, 2002	1.1	Groove version for CygnaCom
September 25, 2002	2.0	CygnaCom initial version
September 30, 2002	2.1	CygnaCom draft version
October 4, 2002	2.2	Name change and CygnaCom version for initial submission to evaluator
October 11, 2002	2.3	Completion of rationale in Section 8
December 4, 2002	2.4	Responded to EORs.
December 7, 2002	2.5	Modified TOE environment to include import and export functional requirements. Added detailed mappings of dependencies for iterations of FCS_COP.1.
December 23, 2002	2.6	Updated ST to respond to EORs.
January 6, 2003	2.7	Made minor updates to fix typos and consistency. Deleted key generation of GDSA algorithm.
January 9, 2003	2.8	Made minor updates to fix typos and redefined some FCS_COP iterations to take out key size ranges.
April 8, 2003	2.9	Made modifications in response to EORs: EOR_ST_01 and EOR_ST_02
April 9, 2003	3.0	Made modifications to respond to minor validator corrections
April 16, 2003	3.09	Made minor modifications to respond to validator comments
April 29, 2003	3.1	Made minor modifications to respond to validator comments
August 11, 2003	3.2	Modified the ST to be consistent with the new TOE definition presented in evaluation evidence.
August 15, 2003	3.3	Added a table of interpretations, deleted the text of the assurance requirements, and made minor corrections to text in Sections 2, 5, and 6.
August 22, 2003	3.4	OE.DESTRUCT was modified to refer to the environment instead of the TOE.

1 SECURITY TARGET INTRODUCTION

1.1 SECURITY TARGET IDENTIFICATION

TOE Identification: Groove Cryptographic Services (GrooveMisc.dll 2.5.0.1774; cryptopp.dll 5.0.4.0)

ST Identification: Groove Cryptographic Services (GrooveMisc.dll 2.5.0.1774; cryptopp.dll 5.0.4.0) Common Criteria Security Target.

ST Version Number: Version 3.4

ST Author: Jean Petty, CygnaCom Solutions, Inc.

Assurance level: EAL2, augmented.

Registration: <To be filled in upon registration>

Keywords: PC, Cryptographic Module, Dynamic Link Library, Collaboration Software.

1.2 SECURITY TARGET OVERVIEW

The Groove software TOE (target of evaluation) consists of software (binary executable code). It provides cryptographic services, and certain non-cryptographic support services, for use by applications, such as Groove collaborative computing software, in performing a variety of operations on PCs running either the Windows 2000, the Windows NT, or the Windows XP operating system. It supports various cryptographic security functions such as:

- Generate symmetric keys for use with various cryptographic algorithms and security functions.
- Generate asymmetric keys for use with various public key cryptographic algorithms.
- Perform various symmetric and asymmetric encryption, digital signature, and key agreement operations using the generated symmetric and asymmetric keys.
- Perform secure hash operations using SHA-1.
- Generate and verify message authentication codes using the FIPS-approved HMAC algorithm.

CygnaCom Solutions, an Entrust Company, developed this ST under contract with Groove. The ST revision history is provided in the front matter of this document.

1.3 COMMON CRITERIA CONFORMANCE

This ST has been built with Common Criteria (CC) Version 2.1 (ISO/IEC 15408 Evaluation Criteria for Information Technology Security; Part 1: Introduction and general model, Part 2: Security functional requirements, and Part 3: Security assurance requirements).

This ST is CC Version 2.1, Part 2 extended, and Part 3 conformant, at Evaluation Assurance Level 2 with Augmentation. EAL2 was augmented with ADV_SPM.1, Informal TOE security policy model.

This ST conforms with the NIAP and International interpretations listed in the front matter of this document.

1.4 RELATED DOCUMENTS

- FIPS 140-2, Security Requirements for Cryptographic Modules, 25 May 2002 http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.
- National Institute of Standards and Technology, *Derived Test Requirements for FIPS PUB 140-2, Security Requirements for Cryptographic Modules*, Draft, November 15, 2001.
- FIPS 197 *Advanced Encryption Standard (AES)*
- FIPS 46-3 *Data Encryption Standard (DES)*
- National Institute of Standards and Technology, *Digital Signature Standard (DSS)*, FIPS 186-2, October 5, 2001.
- American Bankers Association, *Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA),* ANSI X9.31-1998.
- American Bankers Association, *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA),* ANSI X9.62-1998.
- National Institute of Standards and Technology, *Secure Hash Standard*, FIPS 180-1, April 17, 1995.
- National Institute of Standards and Technology, *Keyed-Hash Message Authentication Code (HMAC)* FIPS PUB 198, issued March 6, 2002.
- IEEE *Standard Specifications for Public Key Cryptography*, IEEE 1363-2000.

2 TOE DESCRIPTION

2.1 PRODUCT TYPE

The TOE is software that is loaded on a PC running under the Microsoft Windows 2000, the Windows NT, or the Windows XP operating systems. It provides cryptographic services, and certain non-cryptographic support services, for use by applications (particularly Groove software) in performing a variety of operations.

2.2 GENERAL TOE FUNCTIONALITY

The Groove software TOE (target of evaluation) consists of software (binary executable code). It provides cryptographic services, and certain non-cryptographic support services, for use by applications in performing a variety of operations such as:

- Generate symmetric keys for use with various cryptographic algorithms and security functions.
- Generate asymmetric keys for use with various public key cryptographic algorithms.
- Perform various symmetric and asymmetric encryption, digital signature, and key agreement operations using the generated symmetric and asymmetric keys.
- Perform secure hash operations using SHA-1.
- Generate and verify message authentication codes using the FIPS-approved HMAC algorithm.

A diagram of the Groove TOE and the environment in which it exists is provided in Figure 1 and is explained in the text following.

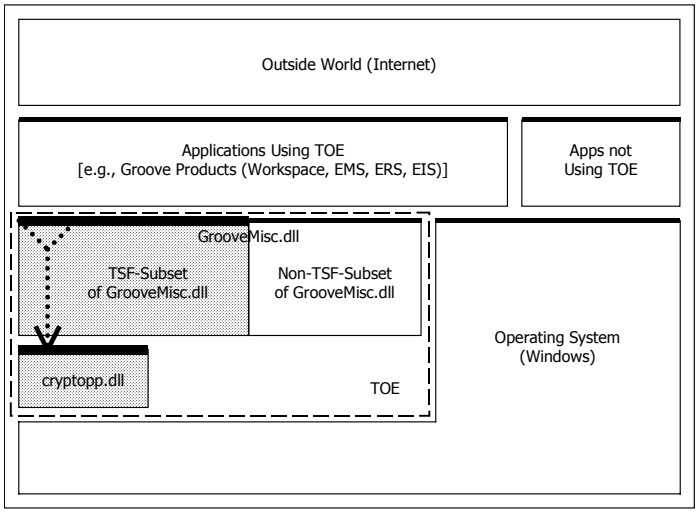

Figure 1. TOE/TSF Diagram

In Figure 1, the boxes indicate software functional components, which provide and consume services amongst one another. The lower-level components in Figure 1 represent providers of services, and higher-level components represent consumers of those services. Services interfaces between producers and consumers are indicated by *heavy horizontal lines*. In general, these interfaces can be APIs (Application Programming Interfaces), or IPCs (Inter-Process Communications).

As shown in Figure 1, the TOE consists of two DLLs (dynamically linked libraries). These DLLs are known as GrooveMisc.dll and cryptopp.dll. The two software DLLs represent the physical boundaries of the TOE. The TOE logical boundaries are depicted in Figure 1 and are described in the following subsections.

The operational environment in which the TOE exists (Microsoft Windows operating system) must provide identification, authentication and access control services sufficient to protect the TOE software from compromise by users.

2.2.1 TSF, TSF Subsystems, and TSFI

The TSF (TOE Security Functions) consists of two subsystems. The two subsystems comprising the TSF are indicated by the shaded portion of Figure 1. The other components of Figure 1 (the non-shaded portions of Figure 1) do not form part of the TSF.

The two subsystems of the TSF are:

- (TSF-Subset of) GrooveMisc.dll — A certain subset of the GrooveMisc.dll DLL (shaded in Figure 1), including the interface to it. *Note:* Where the context ensures that no confusion is likely to result, the qualifying phrase "TSF-Subset of" may be omitted, for simplicity.
- The cryptopp.dll DLL (shaded in Figure 1), including its interface.

As shown in Figure 1, the services of the TSF are accessed by consuming applications via interfaces. These interfaces comprise the TSFI (TSF Interface). Just as the TSF consists of two subsystems, so also the TSFI is composed of two parts, corresponding to the two subsystems of the TSF. One part of the TSFI is the interface accessing the services of the TSF-subset of GrooveMisc.dll. The other part of the TSFI is the interface accessing the services of cryptopp.dll. These two parts of the TSFI are depicted in Figure 1 by the two heaviest horizontal lines, on top of the TSF-subset of GrooveMisc.dll and on top of cryptopp.dll, respectively.

The interfaces of the TSFI are APIs. The binary code implementing these APIs is implemented by exported execution entry points resident in their respective DLLs (they are not separable mechanisms). Applications accessing these APIs must respect the calling conventions associated with these API entry points. These calling conventions (datatypes representing input/output parameters, their encodings, their positions in the computer's execution stack, the location of the associated entry points, etc.) are expressed in the COM and C++ languages.

The interfaces in Figure 1 that access services in the non-TSF subset of GrooveMisc.dll, and the interfaces accessing services outside the TOE, are represented with light-heavy lines, and they do not form part of the TSFI.

2.2.2 GrooveMisc.dll (TSF-Subset) and its COM API

The GrooveMisc.dll DLL consists partially of cryptographic functionality, and partially of non-cryptographic functionality. The cryptographic functionality of GrooveMisc.dll (shaded in Figure 1)

is a subset of the TSF. The non-cryptographic functionality (non-shaded in Figure 1) is not a subset of the TSF (though it is a subset of the TOE).

The TSFI on GrooveMisc.dll consists of APIs expressed in the COM (Common Object Model) interface definition language. COM is considered a "high-level language", which has the capability to specify interfaces only. COM contains no low-level programming constructs, such as arithmetic expressions or execution flow-control operations.

Note that, since the source-code of cryptopp.dll is open-source (see Sec. 2.4.4), it would have been possible to implement all the functionality of the Groove TOE directly in GrooveMisc.dll itself, without splitting off part of the functionality into cryptopp.dll. However, cryptopp.dll is FIPS-validated (see Sec. 2.4.2), while GrooveMisc.dll is not. It is the desire to import the assurance derived from the FIPS-validation of cryptopp.dll into Groove that is the reason for including cryptopp.dll in the Groove TOE.

For the specific cryptographic functionality supported by GrooveMisc.dll, see Sec. 2.5, below.

2.2.3 cryptopp.dll and its C++ API

The cryptopp.dll DLL consists wholly of cryptographic functionality. So it is a subset of the TSF (shaded in Figure 1).

The TSFI on cryptopp.dll consists of APIs expressed in the C++ programming language. C++ is considered a "low-level" language. It has the capability to specify interfaces, and it also supports a complete suite of constructs typically associated with a full-fledged programming language.

The cryptopp.dll DLL contains a set of FIPS-approved cryptographic functionality. It has been validated by NIST (under the title "Crypto++ Library Version 5.0.4") and found to be in conformance with FIPS PUB 140-2 (*Security Requirements for Cryptographic Modules*) Level 1. For details, see the FIPS evaluation documentation for cryptopp.dll, submitted as evidence in cryptopp_august042003.zip.

The cryptopp.dll DLL also runs a suite of self-tests *during initial start-up*, per FIPS specification.

For the specific cryptographic functionality supported by cryptopp.dll, see Sec. 2.5, below.

2.2.4 Relationship Between the Two TSF Subsystems

The cryptopp.dll subsystem is considered to be a minor component of the TSF, subordinate to the major GrooveMisc.dll subsystem, for the reasons explained in this subsection.

In the Groove programming environment, applications are required (by Groove programming convention) to access cryptographic services only via the "high-level" COM APIs on the GrooveMisc.dll TSFI, not via the "low-level" C++ APIs on cryptopp.dll. To implement this convention, some of the COM APIs of the GrooveMisc.dll TSFI act as "wrappers" for the C++ APIs of cryptopp.dll. All of the C++ APIs of cryptopp.dll are "wrapped" (i.e., encapsulated) in this manner. The internal implementation of these high-level "wrapper" COM APIs is to invoke ("wrap") a corresponding low-level C++ API on cryptopp.dll. In this manner, COM APIs on the GrooveMisc.dll can be used to indirectly access services provided by cryptopp.dll. This is indicated in Figure 1 by the *dotted arrow*.

The wrapping strategy just described is necessary to implement Groove's programming convention. Nevertheless, this is only a programming convention. There is no technological enforcement of the

convention. That is, applications can access the C++ APIs of cryptopp.dll if they choose to do so (thereby disregarding the convention). Therefore the C++ APIs on cryptopp.dll must be considered as "external" TSF interfaces for the purposes of this CC evaluation.

The code implementing the wrapping strategy does not modify cryptographic data passing between the COM API TSFI and the underlying C++ API. (Cryptographic data includes cryptographic keys, hash values, message authentication codes, digital signatures, and encrypted or decrypted data.)

COM APIs on the GrooveMisc.dll TSFI are also used to access certain cryptographic services which are implemented directly in GrooveMisc.dll itself. These are "non-wrapper" COM APIs (they do not invoke services in cryptopp.dll).

2.3 CRYPTOGRAPHIC OPERATIONS

The TSF provides Groove cryptographic support by performing the cryptographic operations listed below. FIPS-approved operations are denoted with relevant FIPS publication numbers, and other (non-FIPS) operations are denoted by their relevant specification documents. The operations listed are implemented in the TSF subsystems (GrooveMisc.dll and/or cryptopp.dll), as designated in each subsection below. Those subsystem designations are indicated as:

- Subsystem: GrooveMisc.dll — The operation is implemented entirely in GrooveMisc.dll (cryptopp.dll is not involved). It is externally accessible via a GrooveMisc.dll COM API. (*Note:* Some GrooveMisc.dll COM APIs "wrap" functionality implemented in cryptopp.dll, as explained in Sec. 2.4.3)
- Subsystem: cryptopp.dll — The operation is implemented in cryptopp.dll. It is externally accessible via a cryptopp.dll C++ API. (*Note:* The higher-level COM APIs are the preferred/intended interfaces.)

TOE cryptographic operations include:

- **DES — Data Encryption Standard Key Generation** - The TOE generates 56-bit DES keys (for backward compatibility purposes) as specified in FIPS Publication 46-3. Subsystem: GrooveMisc.dll.
- **DES-ECB — DES Electronic Code Book Encryption and Decryption** - The TOE performs DES encryption and decryption (for backward compatibility purposes) as specified in FIPS Publication 46-3. Subsystem: GrooveMisc.dll.
- **AES — Advanced Encryption Standard Key Generation** - The TOE generates 128, 192, and 256-bit AES keys as specified in FIPS Publication 197. Subsystems: cryptopp.dll.
- **AES-CTR — AES - Counter Mode Encryption and Decryption** - The TOE performs AES encryption and decryption as specified in FIPS Publication 197. Subsystems: cryptopp.dll.
- **DH — Diffie-Hellman Key Generation** - The TOE generates Diffie-Hellman public and private keys of any valid value between 512 bits to 64K bits as defined by the referenced standard IEEE 1363-2000[1]. Subsystem: GrooveMisc.dll.
- **DH — Diffie-Hellman Key Agreement** - The TOE performs Diffie-Hellman key agreement as specified in IEEE 1363-2000. Subsystems: cryptopp.dll.

[1] IEEE 1363-2000 is the IEEE Standard Specifications for Public Key Cryptography.

- **DLIES — Discrete Logarithm Integrated Encryption Scheme Encryption and Decryption** - The TOE performs DLIES encryption and decryption as specified in IEEE P1363a/D11[2]. DLIES encryption and decryption uses Diffie-Hellman public and private keys, generating them as specified in IEEE 1363-2000. Key lengths of 512 to 64K bits are supported. Subsystem: GrooveMisc.dll.
- **RSA — Key Generation** - The TOE generates RSA public and private keys of any valid value between 512 bits to 64K bits as defined by the referenced standard IEEE 1363-2000. Subsystems: cryptopp.dll.
- **RSA — Encryption and Decryption** - The TOE performs RSA encryption and decryption as specified in IEEE 1363-2000. Subsystems: cryptopp.dll.
- **RSA — Signature Generation and Verification** - The TOE performs RSA signature generation and verification as specified in IEEE 1363-2000. Subsystems: cryptopp.dll.
- **GDSA — Generalized DSA Signature Generation and Verification** - The TOE performs GDSA signature generation and verification as specified in the IEEE 1363-2000. GDSA is exactly like DSA except DSA has restrictions on the lower limit for key lengths where 1024 is the minimum key length. GDSA does not have this restriction. GDSA signature generation and verification uses Diffie-Hellman keys as specified in IEEE 1363-2000. Valid key lengths of 512 to 64K bits are supported. Subsystem: GrooveMisc.dll.
- **ESIGN — Key Generation** - The TOE generates ESIGN public and private keys of any valid value between 512 bits to 64K bits as defined by the referenced standard IEEE P1363a/D11. Subsystem: GrooveMisc.dll.
- **ESIGN — Signature Generation and Verification** - The TOE performs ESIGN signature generation and verification as specified IEEE P1363a/D11. Subsystem: GrooveMisc.dll.
- **SHA1 — Secure Hash Algorithm** - The TOE performs SHA1 hash operations as specified in FIPS Publication 180-1. Subsystems: cryptopp.dll.
- **HMAC-SHA1 — Keyed-Hashing for Message Authentication used with SHA1** - The TOE performs HMAC-SHA1 keyed hash operations as specified in FIPS Publication 198. Key lengths of 0 to 0xffffffff (4294967295) bytes are supported. Subsystems: cryptopp.dll.
- **DefaultSecureRandom — FIPS-Approved Random Number Generation** - The TOE performs random number generation as specified in ANSI X9.31. Subsystems: cryptopp.dll.
- **Crypto Module Integrity Test** - The TOE runs a suite of self-tests during initial start-up to verify the integrity of the cryptopp.dll DLL. The self-tests consist of tests specified in FIPS Publication 140-2 for level 1 cryptographic modules. The integrity of the GrooveMisc.dll DLL is not claimed to be checked; however, the trigger for the cryptopp.dll self-test is exposed by the COM API of GrooveMisc.dll. Subsystems: cryptopp.dll.

2.4 ENVIRONMENT SECURITY FUNCTIONAL REQUIREMENTS

The IT environment must provide the following security functional requirements:

- Cryptographic key destruction

[2] IEEE P1363a/D11 is Draft 11 the Proposed IEEE Standard Specifications for Public Key Cryptography: Additional Techniques.

- Identification and authentication
- User data protection
- Security management

All of the TOE security functions (TSF) and security functional requirements for the IT environment are provided by the platform hardware and operating system software on which the TOE executes. The platform has the following hardware requirements:

- Intel® Pentium® processor, 400 MHz or higher
- 64 MB RAM (with 32 MB RAM available for Groove)
- 100 MB free disk space, with additional space required for your data.
- Display resolution 800 x 600, 15-bit (32,768) color minimum

The software (operating system) can be any of the following:

- Microsoft Windows 2000
- Microsoft Windows NT version 4.0 with Service Pack 5 or later
- Microsoft Windows XP

3 SECURITY ENVIRONMENT

This section identifies the following:

- Threats to Security
- Assumptions for the IT environment
- Threats to the Security Environment

3.1 THREATS TO SECURITY

Table 3.1 lists the threats to security.

Table 3.1 – Threats to Security

	Threat Name	Threat Description
1	T.HACK_CRYPTO	Cryptographic algorithms may be incorrectly implemented or may operate incorrectly, allowing an unauthorised individual or user to decipher keys or data and thereby gain unauthorised access to data.
2	T.MALFUNCTION	The TOE may enter an unsecure state at startup due to a malfunction.

3.2 ASSUMPTIONS FOR THE IT ENVIRONMENT

Table 3.2 lists the secure usage assumptions for the IT environment.

Table 3.2 –Assumptions for the IT Environment

	Assumption Name	Assumption Description
1	AE.OS	It is assumed that the TOE is installed on a PC running Microsoft Window 2000, Windows NT version 4.0 with Service Pack 5 or later, or Windows XP.
2	AE.TRUSTED_ADMIN	It is assumed that the administrator, who is responsible for configuring the operating system, is a trusted user and that the administrator will properly install and configure the TOE.

3.3 THREATS TO THE SECURITY ENVIRONMENT

Table 3.3 lists the threats to the IT security environment.

Table 3.3 –Threats to the IT Security Environment

#	Threat Name	Threat Description
1	TE.ATTACK	An undetected compromise of the TOE assets may occur as a result of an attacker (whether an insider or outsider) attempting to perform actions that the individual is not authorised to perform.
2	TE.BYPASS	An unauthorised individual or user may tamper with security attributes or other data in order to bypass OS security functions and gain unauthorised access to TOE assets.
3	TE.CRYPTO_DES	Incorrect cryptographic key destruction may cause an inadvertent disclosure of sensitive information.
4	TE.EXPORT	A user or an attacker may export data to an unsecure location or may export corrupted data, causing the data exported to be added to or substituted for original data and/or to reveal secrets or causing exported data to be erroneous and unusable.
5	TE.IMPERSON	An unauthorised individual may impersonate an authorised user of the OS and thereby gain access to TOE data, keys, and operations.
6	TE.IMPORT	A user or attacker may import data or keys from an unsecure location or data with errors, causing key/data ownership and authorisation to be uncertain or erroneous and/or the system to malfunction or operate in an unsecure manner.
7	TE.MODIFY	An attacker may modify OS or user data, e.g., file permissions, in order to gain access to the TOE and its assets.
8	TE.OBJECT_INIT	An attacker may gain unauthorised access to an object upon its creation if the security attributes are not assigned to the object or an unauthorised individual can assign the security attributes upon object creation.
9	TE.ROLE	A user may assume a more privileged role than permitted and use the enhanced privilege to take unauthorised actions.
10	TE.SECURE_ATT	A user may supply unsecure values for the security attributes of an object and gain unauthorised access to the object.

4 SECURITY OBJECTIVES

4.1 SECURITY OBJECTIVES FOR THE TOE

Table 4.1 lists the security objectives for the TOE.

Table 4.1 – Security Objectives for the TOE

	Objective Name	Objective Description
1	O.ALGORITHMS	The TOE must implement cryptographic algorithms according to specified standards and perform cryptographic operations in accordance with specified algorithms using cryptographic keys of a specified size.
2	O.SELF_TEST	The TOE must perform self-tests at startup to ensure correct functioning of TOE operations.

4.2 SECURITY OBJECTIVES FOR THE ENVIRONMENT

Table 4.2 lists security objectives for the environment.

Table 4.2 – Security Objectives for the Environment

#	Objective Name	Objective Description
1	OE.DAC	The security environment shall control and restrict user access to the TOE assets in accordance with a specified access control policy.
2	OE.DESTRUCT	The security environment must perform cryptographic key destruction in accordance with FIPS Publication 140-2 key destruction requirements.
3	OE.EXPORT	When data are exported outside the TOE, the security environment shall ensure that the data is exported to an authenticated user and location according to the operating system access control rules.
4	OE.I&A	The security environment shall uniquely identify all users and shall authenticate the claimed identity before granting a user access to the TOE facilities.
5	OE.IMPORT	When data are being imported into the TOE, the security environment shall ensure that the data is from authenticated user and location according to the operating system access control rules.
6	OE.INIT_SECURE	The security environment shall provide valid default security attributes when an object is initialized and shall allow only authorised users to change default security attributes.
7	OE.LIMIT_ACTIONS	The security environment shall restrict the actions a user may perform before the TSF verifies the identity of the user.
8	OE.OS	The security environment shall include a PC running Microsoft Windows 2000, Windows NT or Windows XP.
9	OE.SECURE	The security environment shall permit only secure values for security attributes.
10	OE.SECURITY_MGT	The security environment shall enforce access control to ensure that only authorised users may change security attributes.
11	OE.SECURITY_ROLES	The security environment shall maintain security-relevant roles and association of users with those roles.
12	OE.TRUSTED_ADMIN	The security environment shall provide policies and procedures to ensure that the administrator is a trusted user and shall provide functionality that enables the administrator to configure the system in accordance with a specified TOE Security Policy Model.

5 IT SECURITY REQUIREMENTS

5.1 TOE SECURITY FUNCTIONAL REQUIREMENTS

This section contains the security functional requirements (SFRs) for the TOE. The TOE SFRs are Part 2 Extended. The SFRs are Part 2 Extended, because there are explicitly stated requirements as well as requirements drawn from Part 2.

The definition of Part 2 extended is found in the CC Part 3, section 5.4, "Part 2 extended - A PP or TOE is Part 2 extended if the functional requirements include functional components not in Part 2. All TOE functional requirements in this ST are listed in Table 5.1, below. Part 2 extended requirements are explicitly identified as "Part 2 extended."

Table 5.1 – TOE Security Functional Requirements

No.	Component	Component Name	Part 2 or Part 2 extended
1	FCS_CKM.1	Cryptographic key generation	Part 2
2	FCS_COP.1	Cryptographic operation	Part 2
3	FPT_INTTST.1	Crypto Module integrity test	Part 2 extended

The following sections contain the functional components from the Common Criteria (CC) Part 2 with the operations completed and one explicitly stated requirement.

For the Part 2 functional requirements the standard CC text is in regular font; the text inserted by the Security Target (ST) author is in italic font. Iterations of security functional requirements are indicated by the addition of a semicolon followed by the number of the iteration to the component identifier, e.g., FCS_CKM.1;1 and FCS_CKM.1;2 indicate two different iterations of FCS_CKM.1, Cryptographic key generation. Within the component, a semicolon followed by the iteration number is added to each of the requirements, e.g., iteration FCS_CKM.1;1 contains requirement FCS_CKM1.1;1 and iteration FCS_CKM.1;2 contains requirement FCS_CKM1.1;2. Where functional requirement apply to the TSF environment, defined in Section 5.2, functional requirements are refined with the word *environment* in italics.

5.1.1 FCS – Cryptographic Support

FCS_CKM.1;1 Cryptographic key generation - RSA

Hierarchical to: No other components.

FCS_CKM.1.1;1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm *RSA* and specified cryptographic key sizes *any valid value between 512 bits to 64K bits as defined by the referenced standard* that meet the following: *IEEE 1363-2000*.

Dependencies: FCS_COP.1 Cryptographic operation, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes

FCS_CKM.1;2 Cryptographic key generation – AES

Hierarchical to: No other components.

FCS_CKM.1.1;2 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm *Advanced Encryption Standard (AES)* and specified cryptographic key sizes *128, 192, 256 bit* that meet the following: *FIPS PUB 197*.

Dependencies: FCS_COP.1 Cryptographic operation, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes

FCS_CKM.1;3 Cryptographic key generation – DES

Hierarchical to: No other components.

FCS_CKM.1.1;3 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm *Data Encryption Standard (DES)* and specified cryptographic key sizes *56 bits* that meet the following: *FIPS PUB 46-3*.

Dependencies: FCS_COP.1 Cryptographic operation, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes

FCS_CKM.1;4 Cryptographic key generation – DH

Hierarchical to: No other components.

FCS_CKM.1.1;4 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm *Diffie-Hellman* and specified cryptographic key sizes *any valid value between 512 to 64K bits as defined by the referenced standard* that meet the following: *IEEE 1363-2000*.

Dependencies: FCS_COP.1 Cryptographic operation, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes

FCS_CKM.1;5 Cryptographic key generation – ESIGN

Hierarchical to: No other components.

FCS_CKM.1.1;5 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm *ESIGN* and specified cryptographic key sizes *any valid value between 512 bits to 64K bits as defined by the referenced standard* that meet the following: *IEEE 1363a/D11*.

Dependencies: FCS_COP.1 Cryptographic operation, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes

FCS_COP.1;1 Cryptographic operation - RSA encrypt and decrypt

Hierarchical to: No other components.

FCS_COP.1.1;1 The TSF shall perform *encryption and decryption* in accordance with a specified cryptographic algorithm *RSA* and cryptographic key sizes *any valid value as defined by the referenced standard* that meets the following: *IEEE 1363-2000*.

Dependencies: FDP_ITC.1 Import of user data without security attributes, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes.

FCS_COP.1;2 Cryptographic operation - RSA signature and signature verification

Hierarchical to: No other components.

FCS_COP.1.1;2 The TSF shall perform *signature generation and signature verification* in accordance with a specified cryptographic algorithm *RSA* and cryptographic key sizes *any valid value as defined by the referenced standard* that meets the following: IEEE 1363-2000.

Dependencies: FDP_ITC.1 Import of user data without security attributes, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes.

FCS_COP.1;3 Cryptographic operation - SHA-1

Hierarchical to: No other components.

FCS_COP.1.1;3 The TSF shall perform *secure hash* in accordance with a specified cryptographic algorithm *SHA-1* and cryptographic key sizes *not applicable* that meet the following: *FIPS 180-1*.

Dependencies: FDP_ITC.1 Import of user data without security attributes, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes.

FCS_COP.1;4 Cryptographic operation - Keyed-Hashing for Message Authentication

Hierarchical to: No other components.

FCS_COP.1.1;4 The TSF shall perform *keyed-hashing message authentication code (HMAC)* in accordance with a specified cryptographic algorithm *SHA-1* and cryptographic key sizes *160 bits* that meet the following: *FIPS PUB 198*.

Dependencies: FDP_ITC.1 Import of user data without security attributes, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes.

FCS_COP.1;5 Cryptographic operation - DH Key agreement

Hierarchical to: No other components.

FCS_COP.1.1;5 The TSF shall perform *key agreement* in accordance with a specified cryptographic algorithm *Diffie-Hellman* and cryptographic key sizes *any valid value as defined by the referenced standard* that meets the following: *IEEE 1363-2000*.

Dependencies: FDP_ITC.1 Import of user data without security attributes, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes.

FCS_COP.1;6 Cryptographic operation - DES-ECB encrypt and decrypt

Hierarchical to: No other components.

FCS_COP.1.1;6 The TSF shall perform *encryption and decryption* in accordance with a specified cryptographic algorithm *DES Electronic Code Book* and cryptographic key size *56 bits* that meet the following: *FIPS PUB 46-3*.

Dependencies: FDP_ITC.1 Import of user data without security attributes, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes.

FCS_COP.1;7 Cryptographic operation - AES-CTR encrypt and decrypt

Hierarchical to: No other components.

FCS_COP.1.1;7 The TSF shall perform *encryption and decryption* in accordance with a specified cryptographic algorithm *AES Counter Mode* and cryptographic key sizes *128, 192, 256 bits* that meet the following: *FIPS PUB 197*.

Dependencies: FDP_ITC.1 Import of user data without security attributes, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes.

FCS_COP.1;8 Cryptographic operation - GDSA signature and signature verification

Hierarchical to: No other components.

FCS_COP.1.1;8 The TSF shall perform *signature generation and signature verification* in accordance with a specified cryptographic algorithm *GDSA* and cryptographic key sizes *any valid value as defined by the referenced standard* that meets the following: *IEEE 1363-2000*.

Dependencies: FDP_ITC.1 Import of user data without security attributes, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes.

FCS_COP.1;9 Cryptographic operation - DLIES encrypt and decrypt

Hierarchical to: No other components.

FCS_COP.1.1;9 The TSF shall perform *encryption and decryption* in accordance with a specified cryptographic algorithm *Discrete Logarithm Integrated Encryption Scheme (DLIES)* and cryptographic key sizes *any valid value as defined by the referenced standard* that meets the following: *IEEE 1363a/D11*.

Dependencies: FDP_ITC.1 Import of user data without security attributes, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes.

FCS_COP.1;10 Cryptographic operation - ESIGN signature and signature verification

Hierarchical to: No other components.

FCS_COP.1.1;10 The TSF shall perform *signature generation and signature verification* in accordance with a specified cryptographic algorithm *ESIGN* and cryptographic key sizes *any valid value as defined by the referenced standard* that meets the following: *IEEE 1363a/D11*.

Dependencies: FDP_ITC.1 Import of user data without security attributes, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes.

FCS_COP.1;11 Cryptographic operation - RNG

Hierarchical to: No other components.

FCS_COP.1.1;11 The TSF shall perform *random number generation* in accordance with a specified cryptographic algorithm *random number generation* and cryptographic key sizes *not applicable* that meet the following: *ANSI X9.31*.

Dependencies: FDP_ITC.1 Import of user data without security attributes, FCS_CKM.4 Cryptographic key destruction, FMT_MSA.2 Secure security attributes.

5.1.2 FPT – Protection of the TOE Security Functions

FPT_INTTST.1 Crypto module integrity test

Hierarchical to: No other components.

FPT_INTTST.1.1 The TSF shall run a self-test during initial start-up to verify the integrity of the cryptopp.dll.

Dependencies: None

5.2 SECURITY FUNCTIONAL REQUIREMENTS FOR THE ENVIRONMENT

The environment provides the following security functional requirements. These requirements include cryptographic key destruction, identification and authentication, user data protection, and security management functions. All SFRs for the environment are CC Part 2 conformant. Table 5.2 provides a summary list of security functional requirements for the environment.

Table 5.2 – Security Functional Requirements for the Environment

No.	Component	Component Name
1	FCS_CKM.4	Cryptographic key destruction
2	FDP_ACC.1	Subset access control
3	FDP_ACF.1	Security attribute based access control
4	FDP_ETC.1	Export of user data without security attributes
5	FDP_ITC.1	Import of user data without security attributes
6	FIA_UAU.2	User authentication before any action
7	FIA_UID.2	User identification before any action
8	FMT_MSA.1	Management of security attributes
9	FMT_MSA.2	Secure security attributes
10	FMT_MSA.3	Static attribute initialisation
11	FMT_SMF.1	Specification of management functions
12	FMT_SMR.1	Security roles

5.2.1 FCS – Cryptographic Support

FCS_CKM.4 Cryptographic key destruction

Hierarchical to: No other components.

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method *erasure of memory areas containing cryptographic keys* that meets the following: *FIPS 140-2, Key Destruction, or equivalent*.

Dependencies: FCS_CKM.1 Cryptographic key generation, FMT_MSA.2 Secure security attributes

5.2.2 FDP – User Data Protection

FDP_ACC.1 Subset access control

Hierarchical to: No other components.

FDP_ACC.1.1 The TSF *environment* shall enforce the *operating system access controls* on

a) *Subjects: commands executing on behalf of users.*

b) *Objects: keys and data sent to the TOE for a cryptographic operation and from the TOE for storage or use elsewhere.*

c) *Operations: cryptographic operations provided by the TOE.*

Dependencies: FDP_ACF.1 Security attribute based access control

FDP_ACF.1 Security attribute based access control

Hierarchical to: No other components.

FDP_ACF.1.1 The TSF *environment* shall enforce the *operating system access controls* to objects based on *roles and file and data permissions managed by the operating system*.

FDP_ACF.1.2 The TSF *environment* shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: *The roles and permissions as defined by the operating system*.

FDP_ACF.1.3-NIAP-0407 The TSF *environment* shall explicitly authorise access of subjects to objects based on the following additional rules: *No additional rules.*

FDP_ACF.1.4-NIAP-0407 The TSF shall explicitly deny access of subjects to objects based on the *no additional explicit denial rules.*

Dependencies: FDP_ACC.1 Subset access control; FMT_MSA.3 Static attribute initialisation

FDP_ETC.1 Export of user data without security attributes

Hierarchical to: No other components.

FDP_ETC.1.1 The TSF *environment* shall enforce the *operating system access controls* 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.

Dependencies: FDP_ACC.1 Subset access control.

FDP_ITC.1 Import of user data without security attributes

Hierarchical to: No other components.

FDP_ITC.1.1 The TSF *environment* shall enforce the *operating system access controls* 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: *no additional rules*.

Dependencies: FDP_ACC.1 Subset access control, FMT_MSA.3 Static attribute initialisation.

5.2.3 FIA – Identification and Authentication

FIA_UAU.2 User authentication before any action

Hierarchical to: FIA_UAU.1

FIA_UAU.2.1 The TSF *environment* shall require each user to be successfully authenticated before allowing any other TSF *environment*-mediated actions on behalf of that user.

Dependencies: FIA_UID.1 Timing of identification (met by FIA_UID.2, which is hierarchical to FIA_UID.1)

FIA_UID.2 User identification before any action

Hierarchical to: FIA_UID.1

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

Dependencies: No dependencies.

5.2.4 FMT – Security Management

FMT_MSA.1 Management of security attributes

Hierarchical to: No other components

FMT_MSA.1.1 The TSF *environment* shall enforce the *operating system access controls* to restrict the ability to *modify* the security attributes *file permissions or other mechanisms provided by the operating system to protect security assets* to *the specific user as defined by the operating system*.

Dependencies: FDP_ACC.1 Subset access control; FMT_SMR.1 Security roles, FMT_SMF.1 Specification of management functions.

FMT_MSA.2 Secure security attributes

Hierarchical to: No other components

FMT_MSA.2.1 The TSF *environment* shall ensure that only secure values are accepted for security attributes.

Dependencies: ADV_SPM.1 Informal TOE security policy model; FDP_ACC.1 Subset access control; FMT_MSA.1 Management of security attributes; FMT_SMR.1 Security roles.

FMT_MSA.3 Static attribute initialisation

Hierarchical to: No other components.

FMT_MSA.3.1 The TSF *environment* shall enforce the *operating system access controls* to provide *restrictive* default values for security attributes that are used to enforce the SFP.

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

Dependencies: FMT_MSA.1 Management of security attributes, FMT_SMR.1 Security roles.

FMT_SMF.1 Specification of Management Functions

Hierarchical to: No other components

FMT_SMF.1.1 The TSF *environment* shall be capable of performing the following security management functions: *managing the group of roles that can interact with security attributes*.

Dependencies: No dependencies.

FMT_SMR.1 Security roles

Hierarchical to: No other components.

FMT_SMR.1.1 The TSF *environment* shall maintain the roles: *user, administrator*.

FMT_SMR.1.2 The TSF *environment* shall be able to associate users with roles.

Dependencies: FIA_UID.1 Timing of identification (Met by FIA_UID.2, which is hierarchical to FIA_UID.1)

5.3 STRENGTH OF FUNCTION REQUIREMENT

There is no threat level assigned to the TOE and no SOF claim is made, since TOE functional requirements do not include any mechanisms that have a strength of function claim. Strength of function claims cannot be made for cryptography, including key size and strength of algorithm. No ISO 15408 functional or assurance family (including AVA_SOF) may be used for the purposes of evaluating the strength of cryptographic functions or key sizes used. This is because ISO 15408 specifically does not cover the assessment of cryptographic algorithms and related techniques.

The TOE environment includes identification and authentication, for which a SOF analysis can be performed, however, this is outside the scope of the TOE evaluation.

5.4 TOE SECURITY ASSURANCE REQUIREMENTS

The Security Assurance Requirements for the TOE are the assurance components of Evaluation Assurance Level 2 (EAL2) augmented with ADV_SPM.1, Informal TOE security policy model. The augmentation is necessary because ADV_SPM.1 is a dependency for FMT_MSA.2. None of the assurance components is refined. The assurance components are CC Part 3 conformant and are listed in Table 5.3.

Table 5.3 - Assurance Components

Assurance Class		Assurance Components
Configuration Management	ACM_CAP.2	Configuration Items
Delivery and Operation	ADO_DEL.1	Delivery procedures
	ADO_IGS.1	Installation, generation, and start-up procedures
Development	ADV_FSP.1	Informal functional specification
	ADV_HLD.1	Descriptive high-level design
	ADV_RCR.1	Informal correspondence demonstration
	ADV_SPM.1	Informal TOE security policy model **(augmentation)**
Guidance Documents	AGD_ADM.1	Administrator guidance
	AGD_USR.1	User guidance
Tests	ATE_COV.1	Evidence of coverage
	ATE_FUN.1	Functional testing
	ATE_IND.2	Independent testing – sample
Vulnerability Assessment	AVA_SOF.1	Strength of TOE security function evaluation
	AVA_VLA.1	Developer vulnerability analysis

6 TOE SUMMARY SPECIFICATION

6.1 TOE IT SECURITY FUNCTIONS

This section defines the security mechanisms within the TOE that satisfy the functional requirements defined in Section 5. The TOE provides the following security mechanisms: key generation, cryptographic operations, and self test functionality. The functionality is described below, including identification of the particular functional requirement(s) that is met by the functionality. Mapping of functionality to functional requirements is included in Section 8 of this ST.

The TSF provides Groove cryptographic support by performing the cryptographic operations listed below. FIPS-approved operations are denoted with relevant FIPS publication numbers, and other (non-FIPS) operations are denoted by their relevant specification documents. The operations listed are implemented in the TSF subsystems (GrooveMisc.dll and/or cryptopp.dll), as designated in each subsection below. Those subsystem designations are indicated as:

- Subsystem: GrooveMisc.dll — The operation is implemented entirely in GrooveMisc.dll (cryptopp.dll is not involved). It is externally accessible via a GrooveMisc.dll COM API. (*Note:* Some GrooveMisc.dll COM APIs "wrap" functionality implemented in cryptopp.dll, as explained in Sec. 2.4.3)
- Subsystem: cryptopp.dll — The operation is implemented in cryptopp.dll. It is externally accessible via a cryptopp.dll C++ API. (*Note:* The higher-level COM APIs are the preferred/intended interfaces.)

6.1.1 Key Generation

The TOE generates keys for RSA, AES, Diffie-Hellman, DES, and ESIGN. The key generation algorithms are implemented according to the standards listed below and are generated using a FIPS-approved random number generator (specified in ANSI X9.31 – 1998, Appendix A) whenever random numbers are required for key generation:

- **RSA — Key Generation** - The TOE generates RSA public and private keys of any valid value between 512 bits to 64K bits as defined by the referenced standard IEEE 1363-2000. Subsystems: cryptopp.dll.
- **AES — Advanced Encryption Standard Key Generation** - The TOE generates 128, 192, and 256-bit AES keys as specified in FIPS Publication 197. Subsystems: cryptopp.dll.
- **DH — Diffie-Hellman Key Generation** - The TOE generates Diffie-Hellman public and private keys of any valid value between 512 bits to 64K bits as defined by the referenced standard IEEE 1363-2000[3]. Subsystem: GrooveMisc.dll.
- **DES — Data Encryption Standard Key Generation** - The TOE generates 56-bit DES keys (for backward compatibility purposes) as specified in FIPS Publication 46-3. Subsystem: GrooveMisc.dll.

[3] IEEE 1363-2000 is the IEEE Standard Specifications for Public Key Cryptography.

- **ESIGN — Key Generation** - The TOE generates ESIGN public and private keys of any valid value between 512 bits to 64K bits as defined by the referenced standard IEEE P1363a/D11. Subsystem: GrooveMisc.dll.

The TOE key generation capabilities meet the functional requirement FCS_CKM.1, iterated for each algorithm type. A list of the iterations is included below. Note that the name of the subject algorithm is included in the iteration title. Iterations include:

- FCS_CKM.1;1 Cryptographic key generation – RSA
- FCS_CKM.1;2 Cryptographic key generation – AES
- FCS_CKM.1;3 Cryptographic key generation – DES
- FCS_CKM.1;4 Cryptographic key generation – DH
- FCS_CKM.1;5 Cryptographic key generation – ESIGN

6.1.2 Cryptographic Operations

The TOE performs the following cryptographic operations:

- **RSA — Encryption and Decryption** - The TOE performs RSA encryption and decryption as specified in IEEE 1363-2000. Subsystems: cryptopp.dll.
- **RSA — Signature Generation and Verification** - The TOE performs RSA signature generation and verification as specified in IEEE 1363-2000. Subsystems: cryptopp.dll.
- **SHA1 — Secure Hash Algorithm** - The TOE performs SHA1 hash operations as specified in FIPS Publication 180-1. Subsystems: cryptopp.dll.
- **HMAC-SHA1 — Keyed-Hashing for Message Authentication used with SHA1** - The TOE performs HMAC-SHA1 keyed hash operations as specified in FIPS Publication 198. Key lengths of 0 to 0xffffffff (4294967295) bytes are supported. Subsystems: cryptopp.dll.
- **DH — Diffie-Hellman Key Agreement** - The TOE performs Diffie-Hellman key agreement as specified in IEEE 1363-2000. Subsystems: cryptopp.dll.
- **DES-ECB — DES Electronic Code Book Encryption and Decryption** - The TOE performs DES encryption and decryption (for backward compatibility purposes) as specified in FIPS Publication 46-3. Subsystem: GrooveMisc.dll.
- **AES-CTR — AES - Counter Mode Encryption and Decryption** - The TOE performs AES encryption and decryption as specified in FIPS Publication 197. Subsystems: cryptopp.dll.
- **GDSA — Generalized DSA Signature Generation and Verification** - The TOE performs GDSA signature generation and verification as specified in the IEEE 1363-2000. GDSA is exactly like DSA except DSA has restrictions on the lower limit for key lengths where 1024 is the minimum key length. GDSA does not have this restriction. GDSA signature generation and verification uses Diffie-Hellman keys as specified in IEEE 1363-2000. Valid key lengths of 512 to 64K bits are supported. Subsystem: GrooveMisc.dll.
- **DLIES — Discrete Logarithm Integrated Encryption Scheme Encryption and Decryption** - The TOE performs DLIES encryption and decryption as specified in IEEE

P1363a/D11[4]. DLIES encryption and decryption uses Diffie-Hellman public and private keys, generating them as specified in IEEE 1363-2000. Key lengths of 512 to 64K bits are supported. Subsystem: GrooveMisc.dll.

- **ESIGN — Signature Generation and Verification** - The TOE performs ESIGN signature generation and verification as specified IEEE P1363a/D11. Subsystem: GrooveMisc.dll.
- **DefaultSecureRandom — FIPS-Approved Random Number Generation** - The TOE performs random number generation as specified in ANSI X9.31. Subsystems: cryptopp.dll.

The cryptographic operations meet iterations of the functional requirement FCS_COP.1. For cryptographic operations that require keys, the keys generated through FCS_CKM.1 may be used or keys may be imported from smart cards, KDCs (key distribution centers), or other sources. In all cases, the operating system (TOE environment) is responsible for implementing protection of keys and data while they are under the control of the operating system. A list of the iterations is included below. Note that the name of the subject algorithm is included in the iteration title. Iterations include:

- FCS_COP.1;1 Cryptographic operation – RSA encrypt and decrypt
- FCS_COP.1;2 Cryptographic operation – RSA signature and signature verification
- FCS_COP.1;3 Cryptographic operation – SHA-1
- FCS_COP.1;4 Cryptographic operation – Keyed-Hashing for Message Authentication
- FCS_COP.1;5 Cryptographic operation – DH Key agreement
- FCS_COP.1;6 Cryptographic operation – DES-ECB encrypt and decrypt
- FCS_COP.1;7 Cryptographic operation – AES-CTR encrypt and decrypt
- FCS_COP.1;8 Cryptographic operation – GDSA signature and signature verification
- FCS_COP.1;9 Cryptographic operation – DLIES encrypt and decrypt
- FCS_COP.1;10 Cryptographic operation – ESIGN signature and signature verification
- FCS_COP.1;11 Cryptographic operation – RNG

6.1.3 Self-Tests

The TOE runs a suite of self-tests during initial start-up to verify the integrity of the cryptopp.dll DLL. The self-tests consist of tests specified in FIPS Publication 140-2 for level 1 cryptographic modules. These include:

- *Cryptographic algorithm test*. A cryptographic algorithm test using a known answer is conducted for all cryptographic functions (e.g., encryption, decryption, authentication, and random number generation) of each Approved cryptographic algorithm implemented by a cryptographic module. 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 fails.

[4] IEEE P1363a/D11 is Draft 11 the Proposed IEEE Standard Specifications for Public Key Cryptography: Additional Techniques.

- *Software/firmware integrity test*. A software/firmware integrity test using an error detection code (EDC) or Approved authentication technique (e.g., an Approved message authentication code or digital signature algorithm) is applied to all components within the cryptopp.dll on power up.

The integrity of the GrooveMisc.dll DLL is not claimed to be checked; however, the trigger for the cryptopp.dll self-test is exposed by the COM API of GrooveMisc.dll. Subsystems: cryptopp.dll.

The power-up self tests and conditional tests meet the functional requirement explicitly stated requirement FPT_INTTST.1.

The self-test executes the first time the program obtains a pointer to the crypto helper as shown in the following example:

```
IGrooveCryptoHelperPtr pCryptoHelper(CLSID_GrooveCryptoServices);
```

Note that `IGrooveCryptoHelperPtr` is a smart pointer. If object creation fails because self-test failed, the smart pointer throws the exception, enters an error state, and displays an error message:

```
The Groove default cryptographic services provider failed its startup self-test.
Groove may not be properly installed.
```

6.2 ENVIRONMENTAL SECURITY FUNCTIONS

The IT security environment provides significant support to the TOE by providing key destruction, access control, identification and authentication, and security management functionality. The IT security environment is provided by one of the three tested PC operating systems, including the Microsoft Windows 2000, Windows NT, or Windows XP operating systems. The functionality is described below, including identification of the particular functional requirement(s) that is met by the functionality. Mapping of functionality to functional requirements is included in Section 8 of this ST.

6.2.1 Cryptographic Key Destruction

The TOE only stores keys in memory while they are in use. However, the operating system may swap memory that contains keys out to the disk. To zeroize those keys, the swap file must be wiped. One way to do that is to reformat the hard drive(s) containing the swap file, a function that is performed by the operating system. This is the method of key destruction used by the environment. Compliance with the FIPS 140-2 key zeroization requirements was certified as part of the FIPS validation of cryptopp.dll. The environment key destruction capability meets functional requirement FCS_CKM.4. Relevant instructions are provided to Windows administrators in a separate document titled *Installing Groove Workspace in its NIAP-Validated Configuration*.

6.2.2 Access Control

The TOE is restricted to use on Windows 2000, Windows NT, or Windows XP. Setup of the operating system for secure operation is defined in the Administrator Guide for the TOE.

Access control is required to protect sensitive information and operations. FDP_ACC.1 and FDP_ACF.1 require the enforcement of system access controls on all system users for data and operations performed on that data. Note that the Administrator is warned to ignore the periodic “update available” message that appears to notify the user that an update is available; the CC evaluated version should not be updated automatically.

6.2.3 Data and Key Export

The TOE is restricted to use on Windows 2000, Windows NT, or Windows XP. The operating system is responsible for ensuring that data may only be exported from the TOE to a location that is secure and is under the control of the authenticated user. Data and key export is used to move keys from a key generation operation or from a cryptographic operation within the TOE to a user controlled memory or storage location.

Setup of the operating system for secure operation is defined in the Administrator Guide for the TOE.

Data and key export restrictions are required to protect keys and data from unauthorised storage or use. FDP_ETC.1 requires that the environment enforce the operating system access control rules when exporting user data outside of the TOE.

6.2.4 Data and Key Import

The TOE is restricted to use on Windows 2000, Windows NT, or Windows XP. The operating system is responsible for ensuring that data may only be imported to the TOE from a location that is secure and is under the control of the authenticated user. Examples of imported data may be keys, data for hashing, data to be digitally signed, data to be verified, data to be encrypted, data to be decrypted, seed data for random number generation or some combination of several of the above elements. For cryptographic operations that require keys, the keys generated through FCS_CKM.1 may be used or keys may be imported from smart cards, KDCs (key distribution centers), or other sources. In all cases, the operating system is responsible for implementing protection of keys and data while they are under the control of the operating system or the application.

Standard installation of the specified Windows operating system software as defined in the Microsoft Windows administration documentation will ensure that data may only be imported to the TOE from a location that is secure and is under the control of the authenticated user. This is mentioned to TOE administrators in a separate document titled *Installing Groove Workspace in its NIAP-Validated Configuration*.

Data and key export restrictions are required to protect keys and data from unauthorised storage or use. FDP_ITC.1 requires that the environment enforce the operating system access control rules when importing user data from outside of the TOE.

6.2.5 Identification and Authentication

The operating system is responsible for requiring identification and authentication for all users. Windows 2000, Windows NT and Windows XP require a user ID and password for identification and authentication for user access to any system assets or operations, including access to the TOE. This functionality meets functional requirements FIA_UAU.2 and FIA_UID.2. TOE administrator information and relevant procedures are provided to TOE administrators in a separate document titled *Installing Groove Workspace in its NIAP-Validated Configuration*.

6.2.6 Security Management

All persistent security-related data such as keys, passwords, and security attributes reside in the OS file system in an encrypted state to prevent disclosure. Persistent storage and protection of this data is the responsibility of the operating system. OS protection of security attributes and assets meets functional requirements FMT_MSA.1, FMT_MSA.2, FMT_MSA.3, and FMT_SMF.1.

The cryptographic module runs on the Microsoft Windows 2000, Windows NT, or Windows XP Operating System and depends on the operating system for operator identification and authentication. The environment supports the user and administrator roles. The support of the user and administrator roles meets functional requirement FMT_SMR.1. Note that the Administrator is warned to ignore the periodic "update available" message that appears to notify the user that an update is available; the CC evaluated version should not be updated automatically. Relevant environment configuration instructions are provided to Windows administrators in a separate document titled *Installing Groove Workspace in its NIAP-Validated Configuration*.

6.3 STRENGTH OF FUNCTION REQUIREMENT

There is no threat level assigned to the TOE and no SOF claim is made, since TOE functional requirements do not include any mechanisms that have a strength of function claim. Strength of function claims cannot be made for cryptography, including key size and strength of algorithm. No ISO 15408 functional or assurance family (including AVA_SOF) may be used for the purposes of evaluating the strength of cryptographic functions or key sizes used. This is because ISO 15408 specifically does not cover the assessment of cryptographic algorithms and related techniques.

The TOE environment includes identification and authentication, for which a SOF analysis can be performed, however, this is outside the scope of the TOE evaluation.

6.4 ASSURANCE MEASURES

The assurance level selected for the TOE was EAL2 because it is applicable in those circumstances where developers or users require a low to moderate level of independently assured security in the absence of ready availability of the complete development record.

EAL2 also provides assurance through a configuration list for the TOE, and evidence of secure delivery procedures.

This EAL represents a meaningful increase in assurance from EAL1 by requiring developer testing, a vulnerability analysis, and independent testing based upon more detailed TOE specifications.

Appropriate assurance measures will be employed to satisfy the security assurance requirements. The evaluation will confirm whether the assurance measures are sufficient to satisfy the assurance requirements. The assurance measures will consist of the set of evaluation evidence listed in Table 6.1, below. The documents listed in the table will be used as to satisfy assurance evaluation requirements.

Table 6.1 – Assurance Evaluation Evidence

Assurance Requirement	Evidence
ACM_CAP.2	Classes: ACM, ADO, AGD, and AVA, Revision 1.3 (NIAP_ACM_ADO_AGD_AVA)
ADO_DEL.1	Classes: ACM, ADO, AGD, and AVA, Revision 1.3 (NIAP_ACM_ADO_AGD_AVA)
ADO_IGS.1	Classes: ACM, ADO, AGD, and AVA, Revision 1.3 (NIAP_ACM_ADO_AGD_AVA)
ADV_FSP.1	Class ADV: Development Documentation, Revision 1.3 (NIAP_ADV)
ADV_HLD.1	Class ADV: Development Documentation, Revision 1.3 (NIAP_ADV)
ADV_RCR.1	Class ADV: Development Documentation, Revision 1.3 (NIAP_ADV)
ADV_SPM.1	Class ADV: Development Documentation, Revision 1.3 (NIAP_ADV)
AGD_ADM.1	Classes: ACM, ADO, AGD, and AVA, Revision 1.3 (NIAP_ACM_ADO_AGD_AVA)
AGD_USR.1	Classes: ACM, ADO, AGD, and AVA, Revision 1.3 (NIAP_ACM_ADO_AGD_AVA)
ATE_COV.1	Class ATE: Test Documents, Revision 1.3 (NIAP_ATE)
ATE_FUN.1	Class ATE: Test Documents, Revision 1.3 (NIAP_ATE)
ATE_IND.2	Class ATE: Test Documents, Revision 1.3 (NIAP_ATE)
AVA_SOF.1	Classes: ACM, ADO, AGD, and AVA, Revision 1.3 (NIAP_ACM_ADO_AGD_AVA)
AVA_VLA.1	Classes: ACM, ADO, AGD, and AVA, Revision 1.3 (NIAP_ACM_ADO_AGD_AVA)

7 PP CLAIMS

This Security Target was not written to address any existing Protection Profile.

8 RATIONALE

8.1 SECURITY OBJECTIVES RATIONALE

This section consists of two subsections. Section 8.1.1 shows that all of the secure usage assumptions and threats to security have been addressed. Section 8.1.2 shows that each IT security objective and each non-IT security objective counters at least one assumption or threat.

8.1.1 All Assumptions and Threats Addressed

Table 8.1 shows that all the identified assumptions and threats to security have been addressed. Note that TOE assumptions and threats have the prefix “A.” and “T.” respectively. Assumptions and threats for the environment have the prefix “AE.” and “TE.” respectively. The rationale for these mappings is discussed below.

T.HACK_CRYPTO states that cryptographic algorithms may be incorrectly implemented or may operate incorrectly, allowing an unauthorised individual or user to decipher keys or data and thereby gain unauthorised access to data. This threat is countered by O.ALGORITHMS, which ensures that the TOE implements cryptographic algorithms according to specified standards and performs cryptographic operations in accordance with specified algorithms using cryptographic keys of a specified size.

T.MALFUNCTION states that the TOE may enter an unsecure state at startup due to a malfunction. This threat is countered by O.SELF_TEST, which ensures that the TOE performs self-tests at startup to ensure correct functioning of TOE operations.

AE.OS provides the assumption that the TOE is installed on a PC running Microsoft Window 2000, Windows NT or Windows XP. This environmental assumption is met by OE.OS, which ensures that the security environment includes a PC running Microsoft Windows 2000, NT, or XP.

AE.TRUSTED_ADMIN provides the assumption that the Administrator, who is responsible for configuring the operating system, is a trusted user and that the Administrator will properly install and configure the TOE. This environmental assumption is met by OE.TRUSTED_ADMIN, which ensures that appropriate policies and procedures are provided to ensure that the Administrator is a trusted user and that the environment provides functionality that enables an the Administrator to configure the system in accordance with a specified TOE Security Policy Model. This may include operating system configuration and setup manuals and instructions.

TE.ATTACK states that an undetected compromise of the TOE assets may occur as a result of an attacker (whether an insider or outsider) attempting to perform actions that the individual is not authorised to perform. This environmental threat is countered by OE.DAC, which ensures that the security environment controls and restricts user access to the TOE assets in accordance with a specified access control policy.

Table 8.1 – All Assumptions and Threats to Security Countered by Objectives

	Threat Name	Threat Description	Objective
1	T.HACK_CRYPTO	Cryptographic algorithms may be incorrectly implemented or may operate incorrectly, allowing an unauthorised individual or user to decipher keys or data and thereby gain unauthorised access to data.	O.ALGORITHMS
2	T.MALFUNCTION	The TOE may enter an unsecure state at startup due to a malfunction.	O.SELF_TEST
3	AE.OS	It is assumed that the TOE is installed on a PC running Microsoft Window 2000, Windows NT or Windows XP.	OE.OS
4	AE.TRUSTED_ADMIN	It is assumed that the administrator, who is responsible for configuring the operating system, is a trusted user and that the administrator will properly install and configure the TOE.	OE.TRUSTED_ADMIN
5	TE.ATTACK	An undetected compromise of the TOE assets may occur as a result of an attacker (whether an insider or outsider) attempting to perform actions that the individual is not authorised to perform.	OE.DAC
6	TE.BYPASS	An unauthorised individual or user may tamper with security attributes or other data in order to bypass OS security functions and gain unauthorised access to TOE assets.	OE.LIMIT_ACTIONS
7	TE.CRYPTO_DES	Incorrect cryptographic key destruction may cause an inadvertent disclosure of sensitive information.	OE.DESTRUCT
8	TE.EXPORT	A user or an attacker may export data to an unsecure location or may export corrupted data, causing the data exported to be added to or substituted for original data and/or to reveal secrets or causing exported data to be erroneous and unusable.	OE.EXPORT
9	TE.IMPERSON	An unauthorised individual may impersonate an authorised user of the OS and thereby gain access to TOE data, keys, and operations.	OE.I&A
10	TE.IMPORT	A user or attacker may import data or keys from an unsecure location or data with errors, causing key/data ownership and authorisation to be uncertain or erroneous and/or the system to malfunction or operate in an unsecure manner.	OE.IMPORT
11	TE.MODIFY	An attacker may modify OS or user data, e.g., file permissions, in order to gain access to the TOE and its assets.	OE.SECURITY_MGT
12	E.OBJECT_INIT	An attacker may gain unauthorised access to an object upon its creation if the security attributes are not assigned to the object or an unauthorised individual can assign the security attributes upon object creation.	OE.INIT_SECURE
13	TE.ROLE	A user may assume more privileged role than permitted and use the enhanced privilege to take unauthorised actions.	OE.SECURITY_ROLES
14	TE.SECURE_ATT	A user may supply unsecure values for the security attributes of an object and gain unauthorised access to the object.	OE.SECURE

TE.BYPASS states that an unauthorised individual or user may tamper with security attributes or other data in order to bypass OS security functions and gain unauthorised access to TOE assets. This environmental threat is countered by OE.LIMIT_ACTIONS, which ensures that the security environment restricts the actions a user may perform before the TSF verifies the identity of the user.

TE.CRYPTO_DES states that incorrect cryptographic key destruction may cause an inadvertent disclosure of sensitive information. This threat is countered by OE.DESTRUCT, which ensures that the environment performs cryptographic key destruction in accordance with FIPS Publication 140-2 key destruction requirements.

TE.EXPORT states that a user or an attacker may export data to an unsecure location or may export corrupted data, causing the data exported to be added to or substituted for original data and/or to reveal secrets or causing exported data to be erroneous and unusable. This environmental threat is countered by OE.EXPORT, which ensures that when data are exported outside the TOE, the security environment ensures that the data is exported to an authenticated user and location according to the operating system access control rules.

TE.IMPERSON states that an unauthorised individual may impersonate an authorised user of the OS and thereby gain access to TOE data, keys, and operations. This environmental threat is countered by OE.I&A, which ensures that the security environment uniquely identifies all users and authenticates the claimed identity before granting a user access to the TOE facilities.

TE.IMPORT states that a user or attacker may import data or keys from an unsecure location or data with errors, causing key/data ownership and authorisation to be uncertain or erroneous and/or the system to malfunction or operate in an unsecure manner. This environmental threat is countered by OE.IMPORT, which ensures that when data are being imported into the TOE, the security environment ensures that the data is from authenticated user and location according to the operating system access control rules.

TE.MODIFY states that an attacker may modify OS or user data, e.g., file permissions, in order to gain access to the TOE and its assets. This environmental threat is countered by OE.SECURITY_MGT, which ensures that the security environment shall enforce access control to ensure that only authorised users may change security attributes.

TE.OBJECT_INIT states that an attacker may gain unauthorised access to an object upon its creation if the security attributes are not assigned to the object or an unauthorised individual assigns the security attributes upon object creation. This environmental threat is countered by OE.INIT_SECURE, which ensures that the security environment provides valid default security attributes when an object is initialized and that only authorised individuals are allowed to change default security attributes.

TE.ROLE states that a user may assume more privileged role than permitted and use the enhanced privilege to take unauthorised actions. This environmental threat is countered by OE.SECURITY_ROLES, which ensures that the security environment maintains security-relevant roles and association of users with those roles.

TE.SECURE_ATT, A user may supply unsecure values for the security attributes of an object and gain unauthorised access to the object. This environmental threat is countered by OE.SECURE, which ensures that the security environment shall permit only secure values for security attributes.

8.1.2 All Objectives Necessary

Table 8.2 shows that there are no unnecessary IT security objectives for the TOE, since each objective addresses at least one threat or secure usage assumption. Mapping rationale is discussed in the previous section and is not repeated here, since each objective maps to one threat and each threat maps to one objective, i.e., mapping for objectives to threats is the same as threats to objectives.

Table 8.2 – All IT Security Objectives Necessary

	Objective Name	Objective Description	Threat/Assumption
1	O.ALGORITHMS	The TOE must implement cryptographic algorithms according to specified standards and perform cryptographic operations in accordance with specified algorithms using cryptographic keys of a specified size.	T.HACK_CRYPTO
2	O.SELF_TEST	The TOE must perform self-tests at startup to ensure correct functioning of TOE operations.	T.MALFUNCTION
3	OE.DAC	The security environment shall control and restrict user access to the TOE assets in accordance with a specified access control policy.	TE.ATTACK
4	OE.DESTRUCT	The environment must perform cryptographic key destruction in accordance with FIPS 140 key destruction requirements.	TE.CRYPTO_DES
5	OE.EXPORT	When data are exported outside the TOE, the security environment shall ensure that the data is exported to an authenticated user and location according to the operating system access control rules.	TE.EXPORT
6	OE.I&A	The security environment shall uniquely identify all users and shall authenticate the claimed identity before granting a user access to the TOE facilities.	TE.IMPERSON
7	OE.IMPORT	When data are being imported into the TOE, the security environment shall ensure that the data is from authenticated user and location according to the operating system access control rules.	TE.IMPORT
8	OE.INIT_SECURE	The security environment shall provide valid default security attributes when an object is initialized and shall allow only authorised users to change default security attributes.	TE.OBJECT_INIT
9	OE.LIMIT_ACTIONS	The security environment shall restrict the actions a user may perform before the TSF verifies the identity of the user.	TE.BYPASS

Table 8.2, Concluded

	Objective Name	Objective Description	Threat/Assumption
10	OE.OS	The security environment shall include a PC running Microsoft Windows 2000, Windows NT or Windows XP.	AE.OS
11	OE.SECURE	The security environment shall permit only secure values for security attributes.	TE.SECURE_ATT
12	OE.SECURITY_MGT	The security environment shall enforce access control to ensure that only authorised users may change security attributes.	TE.MODIFY
13	OE.SECURITY_ROLES	The security environment shall maintain security-relevant roles and association of users with those roles.	TE.ROLE
14	OE.TRUSTED_ADMIN	The security environment shall provide policies and procedures to ensure that the administrator is a trusted user and shall provide functionality that enables the administrator to configure the system in accordance with a specified TOE Security Policy Model.	AE.TRUSTED_ADMIN

8.2 SECURITY REQUIREMENTS RATIONALE

8.2.1 All Objectives Met by Security Requirements

Table 8.3 maps IT security objectives to functional and assurance requirements. Objectives and mapped requirements for the TOE are identified by an "O." prefix on the objective. Objectives for and mapped requirements for the environment are identified by an "OE." Prefix on the objective. The rationale for the mappings is discussed below.

Table 8.3– Mapping of IT Security Objectives to Requirements

No	Objective Name	Security Requirement
1	O.ALGORITHMS	FCS_CKM.1 FCS_COP.1
2	O.SELF_TEST	FPT_INTTST.1
3	OE.DAC	FDP_ACC.1 FDP_ACF.1
4	OE.DESTRUCT	FCS_CKM.4
5	OE.EXPORT	FDP_ETC.1
6	OE.I&A	FIA_UAU.2 FIA_UID.2
7	OE.IMPORT	FDP_ITC.1
8	OE.INIT_SECURE	FMT_MSA.3
9	OE.LIMIT_ACTIONS	FIA_UAU.2 FIA_UID.2
10	OE.OS	ADO_IGS.1 AGD_ADM.1 ADV_SPM.1
11	OE.SECURE	FMT_MSA.2
12	OE.SECURITY_MGT	FMT_MSA.1 FMT_SMF.1
13	OE.SECURITY_ROLES	FMT_SMR.1 FMT_SMF.1
14	OE.TRUSTED_ADMIN	ADO_IGS.1 AGD_ADM.1 ADV_SPM.1

O.ALGORITHMS states that the TOE must implement cryptographic algorithms according to specified standards and perform cryptographic operations in accordance with specified algorithms using cryptographic keys of a specified size. This objective is met by functional requirements FCS_CKM.1, Cryptographic Key Generation and FCS_COP.1, Cryptographic operation. For key generation, iterations of FCS_CKM.1 define specific industry or government standards that must be met by the TOE. For cryptographic operations such as encrypt, decrypt, signature, signature verification, etc., iterations of FCS_COP.1 define specific industry or government standards that must be met by the TOE.

O.SELF_TEST states that the TOE must perform self-tests at startup to ensure correct functioning of TOE operations. This objective is met by FPT_INTTST.1, Crypto module integrity test. FPT_INTTST.1 requires the performance of a set of self-tests at system startup.

OE.DAC states that the security environment shall control and restrict user access to the TOE assets in accordance with a specified access control policy. This objective is mapped to FDP_ACC.1, Subset access control, and FDP_ACF.1, Security attribute based access control. These two requirements require operating system access control to subjects, objects, and operations based on security attributes.

OE.DESTRUCT states that the environment must perform cryptographic key destruction in accordance with FIPS Publication 140-2 key destruction requirements. This objective is met by FCS_CKM.4, Cryptographic key destruction, which requires that the environment perform key destruction in accordance with FIPS Publication 140-2 key destruction requirements.

OE.EXPORT states that when data are exported outside the TOE, the security environment shall ensure that the data is exported to an authorised user and location according to the operating system access control rules. This objective is met by FDP_ETC.1, Export of user data without security attributes, which requires that the TSF environment enforce the operating system access control SFPs when exporting user data outside the TSC.

OE.I&A states that the security environment shall uniquely identify all users and shall authenticate the claimed identity before granting a user access to the TOE facilities. This objective is met by the functional requirements FIA_UAU.2, User authentication before any action, and FIA_UID.2, User identification before any action. FIA_UAU.2 states that all users must be authenticated prior to any access to the security environment or TOE. FIA_UID.2 states that all users must be identified prior to any access to the security environment or TOE.

OE.IMPORT states that when data are imported into the TOE, the security environment ensures that the data is from an authorised user and location according to the operating system access control rules. This objective is met by FDP_ITC.1, Import of user data without security attributes, which requires that the TSF environment shall enforce the operating system access control SFPs when importing user data from outside the TOE.

OE.INIT_SECURE states that the security environment shall provide valid default security attributes when an object is initialized and that only authorised users are allowed to change default security attributes. This objective is met by FMT_MSA.3, Static attribute initialization, which requires that the operating system provide restrictive default values for security attribute initialization and that the user may override default values with appropriate authorisation.

OE.LIMIT_ACTIONS states that the security environment shall restrict the actions a user may perform before the TSF verifies the identity of the user. This objective is met by the functional requirements FIA_UAU.2, User authentication before any action, and FIA_UID.2, User identification before any action. FIA_UAU.2 states that all users must be authenticated prior to any access to the security environment or actions taken. FIA_UID.2 states that all users must be identified prior to any access to the security environment or actions taken.

OE.OS states that the security environment shall include a PC running Microsoft Windows 2000, Windows NT or Windows XP. This objective is met by assurance requirements ADO_IGS.1, Installation, Generation, and Startup Procedures, AGD_ADM.1, Administrator Guidance, and ADV_SPM.1, Informal TOE Security Policy Model. The first two documents define system installation, setup and administration procedures that ensure that the operating system is Microsoft Windows 2000, Windows NT or Windows XP. ADV_SPM.1 requires a security policy model that states that Microsoft Windows 2000, Windows NT, are Windows XP are the approved operating systems defined for the security policy model.

OE.SECURE states that the security environment shall permit only secure values for security attributes. This objective is met by FMT_MSA.2, Secure security attributes, which requires that only secure values be accepted by the OS for security attributes.

OE.SECURITY_MGT states that the security environment shall enforce access control to ensure that only authorised users may change security attributes. This objective is met by FMT_MSA.1, Management of security attributes, which states that the operating system access control policy shall be enforced to ensure that only authorised users may change security attributes. This objective is supported by FMT_SMF.1, which is a dependency of FMT_MSA.1 that states that the environment must manage the group of roles that are able to change security attributes.

OE.SECURITY_ROLES states that the security environment shall maintain security-relevant roles and association of users with those roles. This objective is met by FMT_SMR.1, Security roles, which defines the roles that must be defined and maintained by the security environment, i.e., user and administrator. This objective is supported by FMT_SMF.1, which states that the environment must manage the group of roles that are able to change security attributes.

O.TRUSTED_ADMIN states that the security environment must provide policies and procedures to ensure that the administrator is a trusted user and that the TOE is properly installed and configured. This objective is met by assurance requirements ADO_IGS.1, Installation, Generation, and Startup Procedures, AGD_ADM.1, Administrator Guidance, and ADV_SPM.1, Informal TOE Security Policy Model. The first two documents define system installation, setup and administration procedures that ensure that the system is configured so that login and password is required prior to any user access, and that security relevant data is properly protected within operating system files. The security policy model defines the data that must be protected, while providing a specific TOE security policy. Documentation on operating system setup and personnel referenced in these manuals provides assurance that there are policies and procedures for ensuring that the administrator is a trusted user.

8.2.2 All Functional Components Necessary

Table 8.4 shows that each functional requirement is necessary, since it is used to address at least one of the IT security objectives. Note that functional requirements for the TOE map to objectives with an “O.” prefix and functional requirements for the environment map to an “OE.” prefix. Discussion of the mapping is provided in the previous section.

Table 8.4– Mapping of Functional Requirements to IT Security Objectives

	Component	Component Name	Objective
1	FCS_CKM.1	Cryptographic key generation	O.ALGORITHMS
2	FCS_CKM.4	Cryptographic key destruction	OE.DESTRUCT
3	FCS_COP.1	Cryptographic operation	O.ALGORITHMS
4	FDP_ACC.1	Subset access control	OE.DAC
5	FDP_ACF.1	Security attribute based access control	OE.DAC
6	FDP_ETC.1	Export of user data without security attributes	OE.EXPORT
7	FDP_ITC.1	Import of user data without security attributes	OE.IMPORT
8	FIA_UAU.2	User authentication before any action	OE.I&A OE.LIMIT_ACTIONS
9	FIA_UID.2	User identification before any action	OE.I&A OE.LIMIT_ACTIONS
10	FMT_MSA.1	Management of security attributes	OE.SECURITY_MGT
11	FMT_MSA.2	Secure security attributes	OE.SECURE
12	FMT_MSA.3	Static attribute initialisation	OE.INIT_SECURE
13	FMT_SMF.1	Specification of management functions	OE.SECURITY_MGT OE.SECURITY_ROLES
14	FMT_SMR.1	Security roles	OE.SECURITY_ROLES
15	FPT_INTTST.1	Crypto module integrity test	O.SELF_TEST

8.2.3 Satisfaction of Dependencies

Table 8.5 shows the dependencies between the functional requirements. Note that dependencies to assurance requirements show a reference of "Assurance" and that these requirements are included in the assurance requirements for the TOE.

Table 8.5 – Functional Requirements Dependencies

No	Component	Component Name	Dependencies	Reference
Functional Requirements for the TOE				
1a	FCS_CKM.1;1	Cryptographic key generation – RSA	FCS_COP.1;1 FCS_COP.1;2 FCS_COP1;11 FCS_CKM.4 FMT_MSA.2	2a 2b 2k 4 12
1b	FCS_CKM.1;2	Cryptographic key generation – AES	FCS_COP.1;7 FCS_CKM.4 FMT_MSA.2	2g 4 12
1c	FCS_CKM.1;3	Cryptographic key generation – DES	FCS_COP.1;6 FCS_CKM.4 FMT_MSA.2	2f 4 12
1e	FCS_CKM.1;4	Cryptographic key generation – DH	FCS_COP.1;5 FCS_COP1;8 FCS_COP.1;9 FCS_COP1;11 FCS_CKM.4 FMT_MSA.2	2e 2h 2i 2k 4 12
1f	FCS_CKM.1;5	Cryptographic key generation - ESIGN	FCS_COP.1;10 FCS_CKM.4 FMT_MSA.2	2j 4 12
2a	FCS_COP.1;1	Cryptographic operation – RSA encrypt and decrypt	FDP_ITC.1 FCS_CKM.4 FMT_MSA.2	8 4 12
2b	FCS_COP.1;2	Cryptographic operation – RSA signature and signature verification	FDP_ITC.1 FCS_CKM.4 FMT_MSA.2	8 4 12
2c	FCS_COP.1;3	Cryptographic operation – SHA-1	FDP_ITC.1 FCS_CKM.4 FMT_MSA.2	8 4 12
2d	FCS_COP.1;4	Cryptographic operation – Keyed-Hashing for Message Authentication	FDP_ITC.1 FCS_CKM.4 FMT_MSA.2	8 4 12
2e	FCS_COP.1;5	Cryptographic operation – DH Key Agreement	FDP_ITC.1 FCS_CKM.4 FMT_MSA.2	8 4 12

No	Component	Component Name	Dependencies	Reference
Functional Requirements for the TOE (concluded)				
2f	FCS_COP.1;6	Cryptographic operation – DES-ECB encrypt and decrypt	FDP_ITC.1 FCS_CKM.4 FMT_MSA.2	8 4 12
2g	FCS_COP.1;7	Cryptographic operation – AES-CTR encrypt and decrypt	FDP_ITC.1 FCS_CKM.4 FMT_MSA.2	8 4 12
2h	FCS_COP.1;8	Cryptographic operation – GDSA signature and signature verification	FDP_ITC.1 FCS_CKM.4 FMT_MSA.2	8 4 12
2i	FCS_COP.1;9	Cryptographic operation – DLIES encrypt and decrypt	FDP_ITC.1 FCS_CKM.4 FMT_MSA.2	8 4 12
2j	FCS_COP.1;10	Cryptographic operation – ESIGN signature and signature verification	FDP_ITC.1 FCS_CKM.4 FMT_MSA.2	8 4 12
2k	FCS_COP.1;11	Cryptographic operation - RNG	FDP_ITC.1 FCS_CKM.4 FMT_MSA.2	8 4 12
3	FPT_INTTST.1	Crypto module integrity test	none	none
Functional Requirements for the Environment				
4	FCS_CKM.4	Cryptographic key destruction	FCS_CKM.1 FMT_MSA.2	1 (all iterations) 12
5	FDP_ACC.1	Subset access control	FDP_ACF.1	6
6	FDP_ACF.1	Security attribute based access control	FDP_ACC.1 FMT_MSA.3	5 13
7	FDP_ETC.1	Export of user data without security attributes	FDP_ACC.1	5
8	FDP_ITC.1	Import of user data without security attributes	FDP_ACC.1 FMT_MSA.3	5 13
9	FIA_UAU.2	User authentication before any action	FIA_UID.1 (met by FIA_UID.2, which is hierarchical to FIA_UID.1)	10
10	FIA_UID.2	User identification before any action	none	none
11	FMT_MSA.1	Management of security attributes	FDP_ACC.1 FMT_SMF.1 FMT_SMR.1	5 14 15
12	FMT_MSA.2	Secure security attributes	ADV_SPM.1 FDP_ACC.1 FMT_MSA.1 FMT_SMR.1	Augmented assurance 5 11 15
13	FMT_MSA.3	Static attribute initialisation	FMT_MSA.1 FMT_SMR.1	11 15
14	FMT_SMF.1	Specification of Management Functions	None	none
15	FMT_SMR.1	Security roles	FIA_UID.1 (met by FIA_UID.2)	10

Note that for dependencies of iterations of FCS_CKM.1 on FCS_COP.1, the dependencies listed are for the iteration of FCS_CKM.1 for key generation that matches the FCS_COP.1 iteration that either provides information for or that uses that key or key pair.

8.2.4 Assurance Rationale

EAL2 was selected as the assurance level because the TOE is a commercial product whose users require a low to moderate level of independently assured security. The TOE has been developed using a cryptographic library for which there is an absence of ready availability of the complete development record. The security objectives defined for the TOE are consistent with an EAL2 assurance level and EAL2 is sufficient to satisfy the security objectives of the TOE.

EAL2 provides assurance by an analysis of the security functions, using a functional and interface specification, guidance documentation and the high-level design of the TOE, to understand the security behaviour. The analysis is supported by independent testing of the TOE security functions, evidence of developer testing based on the functional specification, selective independent confirmation of the developer test results, strength of function analysis, and evidence of a developer search for obvious vulnerabilities (e.g. those in the public domain). EAL2 also provides assurance through a configuration list for the TOE and evidence of secure delivery procedures. The TOE and related documentation have all of the characteristics required for EAL2.

ADV_SPM.1 is an augmentation of EAL2 requirements. It was included because it is a dependency of FMT_MSA.2, which in turn is a dependency of FCS_CKM.4 and FCS_COP.1.

8.3 TOE SUMMARY SPECIFICATION RATIONALE

8.3.1 All TOE Security Functional Requirements Satisfied

Table 8.6 shows that the IT Security Functions in the TOE Summary Specification (TSS) address all of the TOE Security Functional Requirements. The mappings are discussed in detail in Section 6. As described in Section 6, all functional components map to a TOE Security Function. TOE Security functions are:

- Key Generation
- Cryptographic Operations
- Self-tests

Table 8.6 – Mapping of Functional Requirements to TOE Summary Specification

No	Functional Component	Functional Requirement	TOE Security Function
1	FCS_CKM.1	Cryptographic key generation	Key Generation
2	FCS_COP.1	Cryptographic operation	Cryptographic Operations
3	FPT_INTTST.1	Crypto module integrity test	Self-tests

8.3.1.1 Key Generation

The TOE key generation capabilities addressed in Section 6.1.1 meet the functional requirement FCS_CKM.1, iterated for each algorithm type. A list of the iterations is included below. Note that the name of the subject algorithm is included in the iteration title. Iterations include:

- FCS_CKM.1;1 Cryptographic key generation – RSA
- FCS_CKM.1;2 Cryptographic key generation – AES
- FCS_CKM.1;3 Cryptographic key generation – DES
- FCS_CKM.1;4 Cryptographic key generation – DH
- FCS_CKM.1;5 Cryptographic key generation – ESIGN

8.3.1.2 Cryptographic Operations

The cryptographic operations, addressed in Section 6.1.2, meet iterations of the functional requirement FCS_COP.1. A list of the iterations is included below. Note that the name of the subject algorithm is included in the iteration title. Iterations include:

- FCS_COP.1;1 Cryptographic operation – RSA encrypt and decrypt
- FCS_COP.1;2 Cryptographic operation – RSA signature and signature verification
- FCS_COP.1;3 Cryptographic operation – SHA-1
- FCS_COP.1;4 Cryptographic operation – Keyed-Hashing for Message Authentication
- FCS_COP.1;5 Cryptographic operation – DH Key agreement
- FCS_COP.1;6 Cryptographic operation – DES-ECB encrypt and decrypt
- FCS_COP.1;7 Cryptographic operation – AES-CTR encrypt and decrypt
- FCS_COP.1;8 Cryptographic operation – GDSA signature and signature verification
- FCS_COP.1;9 Cryptographic operation – DLIES encrypt and decrypt
- FCS_COP.1;10 Cryptographic operation – ESIGN signature and signature verification
- FCS_COP.1;11 Cryptographic operation – RNG

8.3.1.3 Self-Tests

The power-up self tests and conditional tests, addressed in Section 6.1.3, meet the explicitly stated functional requirement FPT_INTTST.1. The detailed summary specification is not repeated here. Rationale for the explicitly stated requirement is provided in a separate section below.

8.3.2 All Security Functional Requirements for the Environment Satisfied

Table 8.7 shows that the IT Environment Security Functions in the TOE Summary Specification (TSS) address all of the Security Environment Functional Requirements. The mappings are discussed in detail in Section 6. As described in Section 6, all functional components map to an IT Environment Security Function. IT Environment Security functions are:

- Key Destruction
- Access Control
- Data and Key Export

- Data and Key Import
- Identification and authentication
- Security Management

Table 8.7 – Mapping of Functional Requirements to IT Environment Summary Specification

No	Functional Component	Functional Requirement	TOE Security Function
1	FCS_CKM.4	Cryptographic key destruction	Key Destruction
2	FDP_ACC.1	Subset access control	Access Control
3	FDP_ACF.1	Security attribute based access control	Access Control
4	FDP_ETC.1	Export of user data without security attributes	Data and Key Export
5	FDP_ITC.1	Import of user data without security attributes	Data and Key Import
6	FIA_UAU.2	User authentication before any action	Identification and Authentication
7	FIA_UID.2	User identification before any action	Identification and Authentication
8	FMT_MSA.1	Management of security attributes	Security Management
9	FMT_MSA.2	Secure security attributes	Security Management
10	FMT_MSA.3	Static attribute initialisation	Security Management
11	FMT_SMF.1	Specification of Management Functions	Security Management
12	FMT_SMR.1	Security roles	Security Management

8.3.2.1 Cryptographic Key Destruction

The TOE only stores keys in memory while they are in use. However, the operating system may swap memory containing keys to disk. To zeroize those keys, the swap file must be wiped. One way to do that is to reformat the hard drive(s) containing the swap file, a function that is performed by the operating system. This is the method of key destruction used by the environment. The environment key destruction capability meets functional requirement FCS_CKM.4.

8.3.2.2 Access Control

The TOE is restricted to use on Windows 2000, Windows NT, or Windows XP. These operating systems provide access control by limiting access to files and security attributes as defined by the system administrator. Setup of the operating system for secure operation is defined in the Administrator and User Guides and in the Installation, Generation, and Startup Procedures for the TOE.

Access control is required to protect sensitive information and operations. FDP_ACC.1 and FDP_ACF.1 require the enforcement of system access controls on all system users for data and operations performed on that data.

8.3.2.3 Data and Key Export

The TOE is restricted to use on Windows 2000, Windows NT, or Windows XP. The operating system is responsible for ensuring that data may only be exported from the TOE to a location that is secure and is under the control of the authenticated user. Data and key export is used to move keys from a key generation operation or from a cryptographic operation within the TOE to a user controlled memory or storage location.

Setup of the operating system for secure operation is defined in the Administrator and User Guides and in the Installation, Generation, and Startup Procedures for the TOE.

Data and key export restrictions are required to protect keys and data from unauthorised storage or use. FDP_ETC.1 requires that the environment enforce the operating system access control rules when exporting user data outside of the TOE.

8.3.2.4 Data and Key Import

The TOE is restricted to use on Windows 2000, Windows NT, or Windows XP. The operating system is responsible for ensuring that data may only be imported to the TOE from a location that is secure and is under the control of the authenticated user. Examples of imported data may be keys, data for hashing, data to be digitally signed, data to be verified, data to be encrypted, data to be decrypted, seed data for random number generation or some combination of several of the above elements. For cryptographic operations that require keys, the keys generated through FCS_CKM.1 may be used or keys may be imported from smart cards, KDCs (key distribution centers), or other sources. In all cases, the operating system is responsible for implementing protection of keys and data while they are under the control of the operating system or the application.

Setup of the operating system for secure operation is defined in the Administrator and User Guides and in the Installation, Generation, and Startup Procedures for the TOE.

Data and key import restrictions are required to protect keys and data from unauthorised or erroneous import. FDP_ITC.1 requires that the environment enforce the operating system access control rules when importing user data into the TOE.

8.3.2.5 Identification and Authentication

The operating system is responsible for requiring identification and authentication for all users. Windows 2000, Windows NT and Windows XP require a user ID and password for identification and authentication for user access to any system assets or operations, including access to the TOE. This functionality meets functional requirements FIA_UAU.2 and FIA_UID.2.

8.3.2.6 Security Management

All persistent security-related data such as keys, passwords, and security attributes reside in the OS file system in an encrypted state to prevent disclosure. Persistent storage and protection of this data is the responsibility of the operating system. OS protection of security attributes and assets meets functional requirements FMT_MSA.1, FMT_MSA.2, FMT_MSA.3, and FMT_SMF.1.

The cryptographic module runs on the Microsoft Windows 2000, Windows NT, or Windows XP Operating System and depends on the operating system for operator identification and authentication. The environment supports the user and administrator roles. The support of the user and administrator roles meets functional requirement FMT_SMR.1.

8.3.3 All TOE and Environment Summary Specification (TSS) Functions Necessary

Table 8.8 shows that all of the IT Security Functions in the TOE Summary Specification (TSS) and IT Environment Summary Specification are necessary. Explanation and rationale for the mappings are provided in Sections 6.1 and 6.2 and are repeated in Sections 8.3.1 and 8.3.2.

Table 8.8 – Mapping of TOE and Environment Summary Specification to Functional Requirements

TOE/Environment Security Function	Requirement	Requirement Name
Key Generation (TOE)	FCS_CKM.1	Cryptographic key generation
Key Destruction (Environment)	FCS_CKM.4	Cryptographic key destruction
Cryptographic Operations (TOE)	FCS_COP.1	Cryptographic operation
Self-tests (TOE)	FPT_INTTST.1	Crypto module integrity test
Access Control (Environment)	FDP_ACC.1	Subset access control
	FDP_ACF.1	Security attribute based access control
Data and Key Export	FDP_ETC.1	Export of user data without security attributes
Data and Key Import	FDP_ITC.1	Import of user data without security attributes
Identification and Authentication (Environment)	FIA_UAU.2	User authentication before any action
	FIA_UID.2	User identification before any action
Security Management (Environment)	FMT_MSA.1	Management of security attributes
	FMT_MSA.2	Secure security attributes
	FMT_MSA.3	Static attribute initialisation
	FMT_SMR.1	Security roles

8.3.4 Strength of Function Rationale

There is no threat level assigned to the TOE and no SOF claim is made, since TOE functional requirements do not include any mechanisms that have a strength of function claim. Strength of function claims cannot be made for cryptography, including key size and strength of algorithm. No ISO 15408 functional or assurance family (including AVA_SOF) may be used for the purposes of evaluating the strength of cryptographic functions or key sizes used. This is because ISO 15408 specifically does not cover the assessment of cryptographic algorithms and related techniques.

The TOE environment includes identification and authentication, for which a SOF analysis can be performed, however, this is outside the scope of the TOE evaluation.

8.3.5 Rationale for Explicitly Stated Requirement

The explicitly stated requirement FPT_INTTST.1, Crypto module integrity test, is included because no Common Criteria requirement exists that describes the self-test behavior of this TOE. This TOE performs self-tests on only one of the subsystems.

CC requirements FPT_AMT.1 and FPT_TST.1 were considered, however, the TOE does not meet these requirements because FPT_AMT.1 places requirements on the underlying abstract machine, which is not applicable, and FPT_TST.1 applies self test requirements to the entire TOE, which is not applicable.

In order to include the self-test functionality, which is considered important to the security of the TOE, the explicitly stated requirement FPT_INTTST.1 was defined.

8.3.6 Assurance Measures Rationale

The assurance measures rationale shows how all assurance requirements were satisfied. The rationale is provided in Table 8.9.

Table 8.9 – Assurance Evaluation Evidence

Assurance Requirement	Evidence	Rationale
ACM_CAP.2	Classes: ACM, ADO, AGD, and AVA, Version 1.3 (NIAP_ACM_ADO_AGD_AVA)	This evidence was written to address the configuration management documentation for EAL2. This includes identifying the evaluated TOE and providing a configuration list with configuration items that have been uniquely identified and the method used to identify them.
ADO_DEL.1	Classes: ACM, ADO, AGD, and AVA, Version 1.3 (NIAP_ACM_ADO_AGD_AVA)	This evidence addresses delivery procedures for the TOE and documents how the TOE is securely provided to the customer.
ADO_IGS.1	Classes: ACM, ADO, AGD, and AVA, Version 1.3 (NIAP_ACM_ADO_AGD_AVA)	This evidence addresses Installation, Generation, and Startup procedures for the evaluated TOE. This includes that the TOE is installed, generated, and started as the developers intended with the assurance that each time it is done the securely and the same way.
ADV_FSP.1	Class ADV: Development Documentation, Version 1.3 (NIAP_ADV)	This evidence addresses the security functions of the TOE. This includes identifying and describing the external TOE security function interfaces.
ADV_HLD.1	Class ADV: Development Documentation, Version 1.3 (NIAP_ADV)	This evidence describes the security functionality of the TOE and supporting protection mechanisms implemented.
ADV_RCR.1	Class ADV: Development Documentation, Version 1.3 (NIAP_ADV)	This evidence was written specifically to show a correspondence analysis between the ST and the functional specification; between the functional specification and the high level design; and between the functional specification and the security policy model.
ADV_SPM.1	Class ADV: Development Documentation, Version 1.3 (NIAP_ADV)	This evidence provides a security policy model for secure implementation and operation of the TOE.
AGD_ADM.1	Classes: ACM, ADO, AGD, and AVA, Version 1.3 (NIAP_ACM_ADO_AGD_AVA)	This evidence addresses administrator guidance. It describes how to securely administer the TOE and provides references to Operating System documentation.

Table 8.9 (Concluded)

Assurance Requirement	Evidence	Rationale
AGD_USR.1	Classes: ACM, ADO, AGD, and AVA, Revision 1.3 (NIAP_ACM_ADO_AGD_AVA)	This evidence addresses user guidance. It describes the instructions and guidelines for secure use of the TOE and provides references to Operating System documentation.
ATE_COV.1	Class ATE: Test Documents, Revision 1.3 (NIAP_ATE)	This evidence addresses the requirements for test coverage analysis evidence. This includes showing which security functions were tested.
ATE_FUN.1	Class ATE: Test Documents, Revision 1.3 (NIAP_ATE)	This evidence provides the test documentation used by the vendor to test TOE functionality.
ATE_IND.2	Class ATE: Test Documents, Revision 1.3 (NIAP_ATE)	Not applicable – this function is performed and documented by the evaluator and the evidence referenced is used by the evaluator as reference only.
AVA_SOF.1	Classes: ACM, ADO, AGD, and AVA, Revision 1.3 (NIAP_ACM_ADO_AGD_AVA)	This evidence addresses the fact that there are no SOF claims made for the TOE.
AVA_VLA.1	Classes: ACM, ADO, AGD, and AVA, Revision 1.3 (NIAP_ACM_ADO_AGD_AVA)	This evidence addresses the intended environment for the TOE and shows that there are no exploitable obvious vulnerabilities.

8.4 PP CLAIMS RATIONALE

Not applicable.

9 ACRONYMS

CC	Common Criteria for IT Security Evaluation
CM	Configuration Management
EAL	Evaluation Assurance Level
FIPS	Federal Information Processing Standard
SF	Security Function
SFP	Security Function Policy
ST	Security Target
TOE	Target of Evaluation
TSC	TSF Scope of Control
TSF	TOE Security Functions
TSP	TOE Security Policy

10 REFERENCES

Standards

CCITSE	Common Criteria for Information Technology Security Evaluation
FIPS PUB 140-2	Federal Information Processing Standard Publication: Security Requirements for Cryptographic Modules