Network Associates, Inc.

FIPS 140-1

PGP* SDK 1.5 Cryptographic Module Security Policy

Revision 1.4

Table of Contents

Introduction 5

Overview 5

Scope and Purpose of Document 6

Cryptographic Boundary 6

FIPS 140-1 Security Level 6

Roles and Services 7

Security Rules 13

Identification and Authentication Policy 14

Access Control Policy 15

Security Relevant Data Items 15

Accesses 16

Service to SRDI Access Relationship 17

Appendix A 22

Glossary 29

1 Introduction

This document is organized so that the reader understands how the FIPS 140-1 compliance is realized (as it relates to PGP). First, the introduction provides a brief overview of the certification process, the scope and purpose of this document, and the security levels that are being sought for each specific area. From here, the basic roles and services are covered in Section 2. The roles and their services are further defined in a table which lists the services, by category, and their descriptions.

Section 3 provides a listing of the overall security rules that must be used in the design and implementation of the cryptographic module. In section 4, the authentication and identification of the individual operator is addressed. Finally, in section 5, we discuss access to the individual pieces of data. Tables are provided to further define the service-to-data relationship.

1.1 Overview

Network Associates, Inc., is seeking FIPS 140-1 certification for its PGP (Pretty Good Privacy) line of cryptographic software products. PGP software products are used to provide the secure exchange of email and storage of data. This document summarizes the overall security policy for the software within the purview of the FIPS 140-1 standards.

The FIPS 140-1 standard requires vendors of cryptographic products to review more closely the products they create for a variety of security related issues. NIST (National Institute of Standards and Technology) produces a document called the “FIPS PUB 140-1 Security Requirements for Cryptographic Modules,” which sets forth these security issues. It includes guidelines that the FIPS 140-1 certification process requires so that it can be established that these security issues have been properly addressed.

What are these security-related issues and guidelines? The standards require the vendor to establish a physical boundary for what is called a cryptographic module. The FIPS 140-1 standard limits what network connections can be made to and from this physical boundary as well as who will have access to it. The standard requires the vendor to identify operators of the crypto module, their roles (e.g. encryption duties, maintenance, etc.), what functions an operator in a given role can perform, and limitations on the kinds of data to which each operator/role will have access. In so doing, the standard seeks to control the cryptographic module so that data is not corrupted, stolen or compromised.

The vendor must provide supporting documentation to the testing laboratory detailing how these issues have been addressed using the guidelines set forth in FIPS 140-1. At a minimum, FIPS 140-1 requires two documents, the Cryptographic Module Security Policy and a Finite State Model. The Finite State Model details all hardware and software functions and states, effectively mapping any and all data accessed and/or processed. Product-specific documentation identifies these hardware and software functions and describes how they behave.

1.2 Scope and Purpose of Document

This Security Policy specifies the security rules under which the PGP SDK cryptographic module must operate. It includes the rules derived from the security requirements of the FIPS PUB 140-1 standard and rules imposed by Network Associates. These rules define who can access the cryptographic module, how the module can be accessed, and what elements within the module are protected.

1.3 Cryptographic Boundary

The PGP SDK, Version 1.5 is a software-only cryptographic module. To meet the FIPS requirements of a cryptographic module, it is evaluated in its installed state on a Compaq Deskpro 5/166 running Windows NT Workstation, Version 3.51. It is defined as a Multi-chip Stand-alone Module.

The physical Cryptographic Boundary is defined to be the personal computer's case that the PGP SDK, Version 1.5 is installed in. The software Cryptographic Boundary is defined to be a subset of the PGP SDK binary software library. An operator is accessing (or using) the module whenever one of the library calls is executed. Table 6 in “Appendix A” on page 22 lists the PGP SDK calls that are included in the FIPS 140-1 evaluation.

1.4 FIPS 140-1 Security Level

Table 1, “Module Security Level Specification,” on page 6, lists the security levels to which the PGP SDK cryptographic module has been certified.

Table 1: Module Security Level Specification

Security Requirements Section	Level
Cryptographic Module	2
Module Interfaces	2
Roles and Services	2
Finite State Machine	2
Physical Security	2
Software Security	2
Operating System Security	2
Key Management	2
Cryptographic Algorithms	2
EMI/EMC	2

Table 1: Module Security Level Specification

Security Requirements Section	Level
Self Test	2

2 Roles and Services

The cryptographic module supports two roles. An operator accesses both roles while using the PGP SDK. The roles are defined as the following:

- User Role: Shall be allowed to perform services necessary to manage wrapped keys and services necessary to maintain the module.
- Cryptographic Officer Role: Shall be allowed to perform services necessary to generate wrapped keys and services necessary to unwrap keys.

The cryptographic module supports a number of services. The following table lists the services in the cryptographic module organized by service category and indicates which services can be performed by the User Role (indicated by "U" in the Role column) and which by the Cryptographic Officer Role (indicated by a "C" in the Role column). Basically, if a password is required to use a PGP SDK API call, then that service is considered a Cryptographic Officer Role. The abbreviations in this table are used as a reference to these services in other documentation. Appendix A, page 22, contains a table with a mapping of service to PGP SDK API call.

Table 2: Module Services

Service Category	Role	Service	Abbr.	Description
AsymmetricKey Management	U	Open archived key set	K1	Create a new key set and its underlying database based on key data archived in file(s). The key set contains all the keys in the file(s).
	U	Free a key set	K2	Release the storage for a key. The removal includes clearing any memory that held key material.
	U	Import key(s) into key set	K3	Import key(s) that were previously exported into a new key set.
	U	Export key(s) from key set	K4	Export key(s) from a key set into a specified file or buffer.

Table 2: Module Services

Service Category	Role	Service	Abbr.	Description
AsymmetricKey Management (cont.)	U	Archive key set	K5	Writes the key set’s data to a backing store (file) associated with the underlying key database.
	C	Generate a key	K6	Generate a new wrapped PGP key and place it into a key set.
	C	Change key passphrase	K7	Change the passphrase associated with a private key.
	U	Update key properties	K8	Change various values associated with a key including it current trust value, its current status (enabled, disabled, revoked), its user ID, its subkeys, and signatures.
	U	Get key properties	K9	Obtain various values associated with a key including algorithms used, key sizes, user IDs, key IDs and fingerprints, key parameter data, and signature information.
	C	Sign key	K10	Digitally sign a particular key and a specified user ID.
	C	Split a binary passphrase	K11	Split a binary passphrase used to wrap an asymmetric key using the Shamir Threshold Secret Sharing algorithm.
	U	Manage a key set	K12	Operations to create empty key sets, order key references in a key set, and manage the references in a key set.

Table 2: Module Services

Service Category	Role	Service	Abbr.	Description
High-level Cryptographic	U	Encrypt data	C1	Encrypt the provided data with the provided key. This service formats the resultant cipher text based on the OpenPGP Message format.
	C	Sign data	C2	Digitally sign the provided data with the provided key. This service formats the resultant signature based on the OpenPGP Message format.
	C	Decrypt data	C3	Decrypt the provided data with the provided key. This service assumes the data provided is formatted based on the OpenPGP Message format.
	U	Validate signed data	C4	Verify the digital signature on the provided data using the provided key. This service assumes the data provided is formatted based on the OpenPGP Message format.

Table 2: Module Services

Service Category	Role	Service	Abbr.	Description
Low-Level Cryptographic (FIPS parameters must be used to remain in the FIPS mode.)	U	Hash data	L1	Create a hash value based on the provided data.
	U	Compute HMAC on data (non-FIPS approved service)	L2	Compute the message authentication code on the provided data using the provided key.
	U	Encrypt data via symmetric cipher	L3	Encrypt the provided data with the provided key using a symmetric cipher algorithm. The encryption can be of several modes (electronic codebook, cipher block chaining or cipher feedback block).
	U	Decrypt data via symmetric cipher	L4	Decrypt the provided data with the provided key using a symmetric cipher algorithm. The encryption can be of several modes (electronic codebook, cipher block chaining or cipher feedback block).
	U	Encrypt with public key	L5	Encrypt the provided data with the public portion of a public/private key pair.
	U	Verify signature with public key	L6	Verify the digital signature on the provided data using the provided key.
	C	Decrypt with private key	L7	Decrypt the provided data with the private portion of a public/private key pair.
	C	Create signature with private key	L8	Digitally sign the provided data with the provided key.

Table 2: Module Services

Service Category	Role	Service	Abbr.	Description
Random Number	U	Create Random Pool	R1	Create the random pool and initialize with random data.
	U	Get random bytes	R2	Obtain random data.
	U	Get random pool properties	R3	Obtain information about the random pool.
	U	Update random pool	R4	Update the data in the random pool.

Table 2: Module Services

Service Category	Role	Service	Abbr.	Description
Miscellaneous	U	Initialize SDK	M1	Initialize the library for use.
	U	Cleanup SDK	M2	Cleanup the library after use.
	U	Create context	M3	Create a context for a particular use of the library.
	U	Free context	M4	Destroy a context from a particular use of the library.
	U	Data storage management	M5	Create and free memory for holding various data including plaintext and passphrases. For some uses this service is expected to clear any memory that held sensitive data such as passphrases or plaintext.
	U	Data I/O	M6	Deal with buffers and file references for data including plaintext and passphrases.
	U	Option list manipulation	M7	Create, modify, and free a list of options and parameters provided to SDK services.
	U	Module status	M8	Obtain or reset the current status of the PGP SDK cryptographic module.
	U	Run self tests	M9	Run the required self tests.
	U	Get time	M10	Obtain the current time in various formats.
	U	User preferences management	M11	Set up or load preferences used during execution of PGP SDK API calls.

3 Security Rules

The following is a list of the security requirements implemented in the PGP SDK cryptographic module when operating in FIPS mode.

1. The PGP SDK shall provide a FIPS mode of operation.
2. The module shall support the following FIPS approved algorithms:

    - 3DES (encryption / decryption)
    - DSA (digital signature)
    - SHA-1 (hashing)
    - Random Number Generation (ANSI X9.17)

    The module shall support the following algorithms allowed by FIPS:

    - RSA and ElGamal (key distribution)
    - Shamir Threshold Secret Sharing (split knowledge)

    The module shall support the following non-FIPS approved algorithms:

    - CAST5 (encryption / decryption)
    - IDEA (encryption / decryption)
    - RSA (digital signature)
    - MD5 (hashing)
    - RIPEMD60 (hashing)
    - HMAC (data authentication)

3. The PGP SDK shall inhibit all data output via the data output interface whenever an error state exists and during self-tests.
4. The PGP SDK shall support a User role and a Cryptographic Officer role.
5. The module's access control policy is stated in section 5, “Access Control Policy” on page 15 of this document (the SPM).
6. The module's I&A policy is stated in section 4, “Identification and Authentication Policy” on page 14 of this document (the SPM).
7. For a particular invocation of the module software in a process, only one operator is associated with that process (the PGP SDK does not support multiple concurrent operators).
8. The PGP SDK shall be installed on a host computer certified to TCSEC, Level C2 or equivalent specified by NIST Implementation Guidance.

9. The private portion of a DSA public/private key pair shall be passphrase encrypted before export or archive so that it is not disclosed or modifiable.
10. A secret key shall be encrypted before export.
11. The private portion of a DSA public/private key pair shall only be decrypted when needed and when done the cleartext private data is zeroized.
12. The public portion of a DSA public/private key pair shall be digitally signed to protect against unauthorized modifications or substitution.
13. Shared binary passphrases shall only be output by the module after splitting.
14. Upon first use of the toolkit or when commanded by the operator, the module shall perform the following tests:
    - cryptographic algorithm tests (known answer test for DSA sign, DSA verify, encrypt/decrypt 3DES ECB, encrypt/decrypt 3DES CFB, encrypt/decrypt 3DES CBC)
    - software/firmware test (which shall be a digital signature using DSA/SHA-1)
15. If a self-test fails, the module shall enter a FIPS persistent error state and no cryptographic functions will be allowed until reinitialization.
16. The PGP SDK shall not support bypass mode.
17. The PGP SDK shall perform the following conditional tests:
    - prior to each use, the internal random number generator shall be tested using the continuous random number generator test
    - pairwise consistency test on new DSA keys
18. The PGP SDK shall conform, as a minimum, to the EMI/EMC requirements specified in FCC Par 15, Subpart J, Class B.
19. The PGP SDK shall output an error indicator via the status interface whenever an error state is entered due to a failed self-test.

4 Identification and Authentication Policy

For the PGP SDK running on an approved operating system, a module operator must authenticate to the operating system before using the module. Successfully authenticating to the operating system indicates that the operator is authorized to perform the User Role and the Cryptographic Officer Role in the module and the operator is authorized to execute all the services allowed by those roles. An operator does not change roles in the module. If the module is powered down (in other words, if the computer the operating system and module are executing on are powered down), then the operator must reauthenticate to the operating system before using the module.

The module relies on the operating system's I&A mechanism, assuming that the operator must at least provide an identifier and have knowledge of a password associated with that identifier. Once authenticated, it is only that operator who is allowed to use the module.

The cryptographic module then uses role-based authentication to control access to sensitive data (e.g., private key material). Before being allowed to access sensitive data, or to perform a service using the sensitive data, the operator must provide information unique to that operation (individual passphrase). This authenticates the Cryptographic Officer role and permits use of the Cryptographic Officer services.

5 Access Control Policy

In the PGP SDK, access to security relevant data is controlled. A SDK User or Cryptographic Officer cannot read, modify, or otherwise access the security relevant data except through specified cryptographic module services. This section details the security relevant data items (SRDIs) in the cryptographic module that an User or Cryptographic Officer can access, how the SRDIs can be accessed in the cryptographic module, and which services are used for access to the data item.

5.1 Security Relevant Data Items

An operator using the PGP SDK in the User role can access many of the services described in section 2, “Roles and Services” on page 7. An operator in the Cryptographic Officer role can access all of the services. Table 3, “Module Security Relevant Data Items,” on page 15, contains all of the SRDIs in the cryptographic module.

Note: SRDIs such as keys, user IDs and signatures are data items which contain cryptographic data or which are used in secure functionality. The PGP SDK also contains data structures which are "references" or "pointers" to these SRDIs. These reference structures are not considered SRDIs themselves; rather, they form a synopsis of the relationships between SRDIs. Many PGP SDK API calls just deal with reference structures. When this is the case, the API call is not considered to be part of the cryptographic boundary, since the API call is never dealing with an actual SRDI.

Table 3: Module Security Relevant Data Items

SRDI	Description
Asymmetric Key	An asymmetric key contains the actual key material for a public and/or private key.
User ID	An identifier that is associated with an asymmetric key (via a Signature) that represents the entity (e.g., user) to which the key is assigned.
Signature	A value that associates a User ID with an asymmetric key, it represents the binding of key material with a User ID. There can be any number of signatures on a particular key material/User ID pair.
ASCII Passphrase	A sequence of ASCII characters provided by the cryptographic module operator and used to wrap and unwrap private key material.

Table 3: Module Security Relevant Data Items

SRDI	Description
Binary Passphrase	Binary data used in place of an ASCII passphrase to wrap and unwrap private key material.
Symmetric Key	Key material used for symmetric ciphers. A symmetric key can be provided by the operator or created as needed by the PGP SDK.
SymmetricCipher Context	A context which represents the state of a symmetric cipher. The context includes which algorithm is being used, which symmetric cipher mode is being used, a symmetric key, an initialization vector, and plaintext.
Asymmetric Cipher Context	A context which represents the state of an asymmetric cipher. The context includes which algorithm is being used, and either public or private key material.
Hash Context	A context which represents the state of the hash of some data. The context contains the hash algorithm being used and any hash data.
MAC Context	A context which represents the state of the message authenticate code calculation of some data. The context contains the MAC secret and hash data.
Random Pool	An internally maintained pool of data for seeding the random number functions.

5.2 Accesses

The types of access to SRDIs in the PGP SDK are listed in Table 4, “SRDI Access Types,” on page 16.

Table 4: SRDI Access Types

Access	Description
create	the SRDI is created as a result of some service
destroy	the SRDI is destroyed, in other words the SRDI data is cleared from any memory in the cryptographic module and then that memory is released
read	the SRDI data is accessed for reading and use
write	the SRDI data is modified or changed
wrap	the SRDI data is passphrase encrypted with either a binary or ASCII passphrases

Table 4: SRDI Access Types

Access	Description
unwrap	the SRDI data is passphrase decrypted with either a binary or ASCII passphrases

5.3 Service to SRDI Access Relationship

Table 5 on page 17 shows which SRDIs are accessed by each service, the role(s) the operator must be in for access, and how the SRDI is accessed on behalf of the operator when the service is performed. In the table, the letter U represents access by an operator in the User role and the letter C represents access by an operator in the Cryptographic Officer role. Several services provided by the PGP SDK do not access any SRDIs and are included here for completeness.

Table 5: Module Services vs. SRDI Access vs. Role Access

Service	SRDIs	create	destroy	read	write	wrap	unwrap
Open archived key set	Asymmetric Key				U		
	Signature				U		
	UserID				U		
Free a key set	Asymmetric Key		U				
	Signature		U				
	UserID		U				
Import key(s) into key set	Asymmetric Key				U		
	Signature				U		
	UserID				U		
Export key(s) from key set	Asymmetric Key			U			
	Signature			U			
	UserID			U			

Table 5: Module Services vs. SRDI Access vs. Role Access

Service	SRDIs	create	destroy	read	write	wrap	unwrap
Archive key set	Asymmetric Key			U			
	Signature			U			
	UserID			U			
Generate a key	Asymmetric Key	C				C	
	Signature	C					
	UserID	C					
	Passphrase			C			
	Random Seed Pool			C			
Change key passphrase	Asymmetric Key					C	C
	Passphrase			C			
Update key properties	Asymmetric Key			U	U		
Get key properties	Asymmetric Key			U			
Sign key	Asymmetric Key			C			C
	UserID			C			
	Signature	C					
Binary Passphrase Split	Binary passphrase			C			
Manage a key set	N / A						
Encrypt data	Asymmetric Key			U			
	Session Key	U				U	

Table 5: Module Services vs. SRDI Access vs. Role Access

Service	SRDIs	create	destroy	read	write	wrap	unwrap
Sign data	Asymmetric Key			C			C
Decrypt data	Asymmetric Key			C			C
	Session Key			C			C
Validate signed data	Asymmetric Key			U			
Hash data	Hash Context	U	U	U	U		
Compute MAC on data	HMAC Context	U	U	U	U		
Symmetric cipher encrypt	Symm Cipher Context	U	U	U	U		
Symmetric cipher decrypt	Symm Cipher Context	C	C	C	C		
Encrypt with public key	Asymmetric Key			U			
	Asymm Cipher Context	U	U	U	U		
Verify signature with public key	Asymmetric Key			U			
	Signature			U			
	Asymm Cipher Context	U	U	U	U		

Table 5: Module Services vs. SRDI Access vs. Role Access

Service	SRDIs	create	destroy	read	write	wrap	unwrap
Decrypt with private key	Asymmetric Key			C			C
	Passphrase			C			
	Asymm Cipher Context	C	C	C	C		
Create signature with private key	Asymmetric Key			C			C
	Passphrase			C			
	Asymm Cipher Context	C	C	C	C		
Create Random Pool	Random Pool	U					
Get random bytes	Random Pool			U			
Get random pool properties	Random Pool			U			
Update random pool	Random Pool				U		
Initialize SDK	Random Pool	U					
Cleanup SDK	N/A						
Create context	N/A						
Free context	N/A						
Data storage management	N/A						
Data I/O	N/A						
Option list manipulation	Passphrase			C	C		
Module Status	N/A						

Table 5: Module Services vs. SRDI Access vs. Role Access

Service	SRDIs	create	destroy	read	write	wrap	unwrap
Run self tests	N / A						
Get time	N / A						
User Preferences Management	N / A						

Appendix A

Table 6 lists all the PGP SDK API calls that are included in the cryptographic module (CM), which of those are in the cryptographic boundary (CB), which service the API call represents, and the PGP SDK header file that contains the call's prototype.

Table 6: PGP SDK Calls in Cryptographic Module

API Call	Service	CM	CB	Header
PGPCBCDecrypt	L4	Yes	Yes	pgpCBC.h
PGPCBCEncrypt	L3	Yes	Yes	pgpCBC.h
PGPCBCGetSymmetricCipher	L3, L4	Yes	Yes	pgpCBC.h
PGPCopyCBCContext	L3, L4	Yes	Yes	pgpCBC.h
PGPFreeCBCContext	L3, L4	Yes	Yes	pgpCBC.h
PGPInitCBC	L3, L4	Yes	Yes	pgpCBC.h
PGPNewCBCContext	L3, L4	Yes	Yes	pgpCBC.h
PGPCFBDecrypt	L4	Yes	Yes	pgpCFB.h
PGPCFBEncrypt	L3	Yes	Yes	pgpCFB.h
PGPCFBGetRandom	R1	Yes	Yes	pgpCFB.h
PGPCFBGetSymmetricCipher	L3, L4	Yes	Yes	pgpCFB.h
PGPCFBRandomCycle	R3	Yes	Yes	pgpCFB.h
PGPCFBRandomWash	R3	Yes	Yes	pgpCFB.h
PGPCFBSync	L3, L4	Yes	Yes	pgpCFB.h
PGPCopyCFBContext	L3, L4	Yes	Yes	pgpCFB.h
PGPFreeCFBContext	L3, L4	Yes	Yes	pgpCFB.h
PGPInitCFB	L3, L4	Yes	Yes	pgpCFB.h
PGPNewCFBContext	L3, L4	Yes	Yes	pgpCFB.h
PGPAddJobOptions	M7	Yes	Yes	pgpEncode.h
PGPDecode	C3, C4	Yes	Yes	pgpEncode.h
PGPEncode	C1, C2	Yes	Yes	pgpEncode.h
PGPContinueHash	L1	Yes	Yes	pgpHash.h

Table 6: PGP SDK Calls in Cryptographic Module

API Call	Service	CM	CB	Header
PGPCopyHashContext	L1	Yes	Yes	pgpHash.h
PGPFinalizeHash	L1	Yes	Yes	pgpHash.h
PGPFreeHashContext	L1	Yes	Yes	pgpHash.h
PGPGetHashSize	L1	Yes	No	pgpHash.h
PGPNewHashContext	L1	Yes	Yes	pgpHash.h
PGPResetHash	L1	Yes	Yes	pgpHash.h
PGPContinueHMAC	L2	Yes	Yes	pgpHMAC.h
PGPFinalizeHMAC	L2	Yes	Yes	pgpHMAC.h
PGPFreeHMACContext	L2	Yes	Yes	pgpHMAC.h
PGPNewHMACContext	L2	Yes	Yes	pgpHMAC.h
PGPResetHMAC	L2	Yes	Yes	pgpHMAC.h
PGPAddAttributeUserID	K8	Yes	Yes	pgpKeys.h
PGPAddKeyOptions	K8	Yes	Yes	pgpKeys.h
PGPAddKeys	K8	Yes	No	pgpKeys.h
PGPAddUserID	K8	Yes	Yes	pgpKeys.h
PGPChangePassphrase	K7	Yes	Yes	pgpKeys.h
PGPChangeSubKeyPassphrase	K7	Yes	Yes	pgpKeys.h
PGPCheckKeyRingSigs	C4	Yes	Yes	pgpKeys.h
PGPCommitKeyRingChanges	K5	Yes	Yes	pgpKeys.h
PGPCompareKeyIDs	K9	Yes	No	pgpKeys.h
PGPCompareKeys	K9	Yes	No	pgpKeys.h
PGPCountKeys	K9	Yes	No	pgpKeys.h
PGPEnableKey	K8	Yes	No	pgpKeys.h
PGPExportKeyID	K8	Yes	No	pgpKeys.h
PGPExportKeySet	K4	Yes	Yes	pgpKeys.h

Table 6: PGP SDK Calls in Cryptographic Module

API Call	Service	CM	CB	Header
PGPFreeKeyIter	K12	Yes	No	pgpKeys.h
PGPFreeKeyList	K12	Yes	No	pgpKeys.h
PGPFreeKeySet	K2	Yes	Yes	pgpKeys.h
PGPGenerateKey	K6	Yes	Yes	pgpKeys.h
PGPGenerateSubKey	K6	Yes	Yes	pgpKeys.h
PGPGetKeyBoolean	K9	Yes	No	pgpKeys.h
PGPGetKeyByKeyID	K9	Yes	No	pgpKeys.h
PGPGetKeyContext	K9	Yes	No	pgpKeys.h
PGPGetKeyIDFromKey	K9	Yes	No	pgpKeys.h
PGPGetKeyIterContext	K12	Yes	No	pgpKeys.h
PGPGetKeyListContext	K12	Yes	No	pgpKeys.h
PGPGetKeyNumber	K9	Yes	No	pgpKeys.h
PGPGetKeyPasskeyBuffer	K9	Yes	Yes	pgpKeys.h
PGPGetKeySetContext	K12	Yes	No	pgpKeys.h
PGPGetSubKeyPasskeyBuffer	K9	Yes	Yes	pgpKeys.h
PGPGetUserIDNumber	K9	Yes	No	pgpKeys.h
PGPImportKeyID	K8	Yes	No	pgpKeys.h
PGPImportKeySet	K3	Yes	Yes	pgpKeys.h
PGPIncKeyListRefCount	K12	Yes	No	pgpKeys.h
PGPIncKeySetRefCount	K12	Yes	No	pgpKeys.h
PGPKeyIterMove	K12	Yes	No	pgpKeys.h
PGPKeyIterNext	K12	Yes	No	pgpKeys.h
PGPKeySetIsMember	K9	Yes	No	pgpKeys.h
PGPKeySetIsMutable	K9	Yes	No	pgpKeys.h
PGPNewEmptyKeySet	K12	Yes	No	pgpKeys.h

Table 6: PGP SDK Calls in Cryptographic Module

API Call	Service	CM	CB	Header
PGPNewKeyIter	K12	Yes	No	pgpKeys.h
PGPNewKeySet	K12	Yes	No	pgpKeys.h
PGPNewSingletonKeySet	K12	Yes	No	pgpKeys.h
PGPOpenDefaultKeyRings	K1	Yes	Yes	pgpKeys.h
PGPOpenKeyRing	K1	Yes	Yes	pgpKeys.h
PGPOpenKeyRingPair	K1	Yes	Yes	pgpKeys.h
PGPOrderKeySet	K12	Yes	No	pgpKeys.h
PGPPassphraseIsValid	K9	Yes	Yes	pgpKeys.h
PGPRemoveKeys	K2	Yes	No	pgpKeys.h
PGPRemoveSig	K8	Yes	Yes	pgpKeys.h
PGPRemoveSubKey	K2	Yes	Yes	pgpKeys.h
PGPRemoveUserID	K8	Yes	Yes	pgpKeys.h
PGPRevokeKey	K8	Yes	Yes	pgpKeys.h
PGPRevokeSig	K8	Yes	Yes	pgpKeys.h
PGPRevokeSubKey	K8	Yes	Yes	pgpKeys.h
PGPSecretReconstructData	K11	Yes	Yes	pgpKeys.h
PGPSecretShareData	K11	Yes	Yes	pgpKeys.h
PGPSetDefaultPrivateKey	K8	Yes	No	pgpKeys.h
PGPSetKeyAxiomatic	K8	Yes	Yes	pgpKeys.h
PGPSignUserID	K10	Yes	Yes	pgpKeys.h
PGPUnionKeySets	K12	Yes	No	pgpKeys.h
PGPFreeData	M5	Yes	Yes	pgpMemoryMgr.h
PGPFreeMemoryMgr	M5	Yes	No	pgpMemoryMgr.h
PGPGetDefaultMemoryMgr	M5	Yes	No	pgpMemoryMgr.h
PGPGetMemoryMgrCustomValue	M5	Yes	No	pgpMemoryMgr.h

Table 6: PGP SDK Calls in Cryptographic Module

API Call	Service	CM	CB	Header
PGPNewData	M5	Yes	Yes	pgpMemoryMgr.h
PGPNewMemoryMgr	M5	Yes	No	pgpMemoryMgr.h
PGPNewMemoryMgrCustom	M5	Yes	No	pgpMemoryMgr.h
PGPNewSecureData	M5	Yes	Yes	pgpMemoryMgr.h
PGPReallocData	M5	Yes	Yes	pgpMemoryMgr.h
PGPSetDefaultMemoryMgr	M5	Yes	No	pgpMemoryMgr.h
PGPSetMemoryMgrCustomValue	M5	Yes	No	pgpMemoryMgr.h
PGPAppendOptionList	M7	Yes	Yes	pgpOptionList.h
PGPBuildOptionList	M7	Yes	Yes	pgpOptionList.h
PGPCopyOptionList	M7	Yes	Yes	pgpOptionList.h
PGPFreeOptionList	M7	Yes	Yes	pgpOptionList.h
PGPNewOptionList	M7	Yes	Yes	pgpOptionList.h
PGPOAllocatedOutputBuffer	M6	Yes	Yes	pgpOptionList.h
PGPODiscardOutput	M7	Yes	No	pgpOptionList.h
PGPOImportKeysTo	M7	Yes	No	pgpOptionList.h
PGPOInputBuffer	M6	Yes	Yes	pgpOptionList.h
PGPOInputFile	M6	Yes	Yes	pgpOptionList.h
PGPOLastOption	M7	Yes	No	pgpOptionList.h
PGPOOutputBuffer	M6	Yes	Yes	pgpOptionList.h
PGPOOutputFile	M6	Yes	Yes	pgpOptionList.h
PGPOPasskeyBuffer	M6	Yes	Yes	pgpOptionList.h
PGPOPassphrase	M6	Yes	Yes	pgpOptionList.h
PGPOPassphraseBuffer	M6	Yes	Yes	pgpOptionList.h
PGPORawPGPInput	M6	Yes	Yes	pgpOptionList.h
PGPFreePrivateKeyContext	L7, L8	Yes	Yes	pgpPublicKey.h

Table 6: PGP SDK Calls in Cryptographic Module

API Call	Service	CM	CB	Header
PGPFreePublicKeyContext	L5, L6	Yes	Yes	pgpPublicKey.h
PGPGetPrivateKeyOperationSizes	L7, L8	Yes	No	pgpPublicKey.h
PGPGetPublicKeyOperationSizes	L5, L6	Yes	No	pgpPublicKey.h
PGPNewPrivateKeyContext	L7, L8	Yes	Yes	pgpPublicKey.h
PGPNewPublicKeyContext	L5, L6	Yes	Yes	pgpPublicKey.h
PGPPrivateKeyDecrypt	L7	Yes	Yes	pgpPublicKey.h
PGPPrivateKeySign	L8	Yes	Yes	pgpPublicKey.h
PGPPrivateKeySignRaw	L8	Yes	Yes	pgpPublicKey.h
PGPPublicKeyEncrypt	L5	Yes	Yes	pgpPublicKey.h
PGPPublicKeyVerifyRaw	L6	Yes	Yes	pgpPublicKey.h
PGPPublicKeyVerifySignature	L6	Yes	Yes	pgpPublicKey.h
PGPGlobalRandomPoolAddKeystroke	R3	Yes	Yes	pgpRandomPool.h
PGPGlobalRandomPoolAddMouse	R3	Yes	Yes	pgpRandomPool.h
PGPGlobalRandomPoolGetEntropy	R2	Yes	No	pgpRandomPool.h
PGPGlobalRandomPoolGetMinimum Entropy	R2	Yes	No	pgpRandomPool.h
PGPGlobalRandomPoolGetSize	R2	Yes	No	pgpRandomPool.h
PGPGlobalRandomPoolHasMinimum Entropy	R2	Yes	No	pgpRandomPool.h
PGPsdkLoadDefaultPrefs	M11	Yes	No	pgpSDKPrefs.h
PGPsdkLoadPrefs	M11	Yes	No	pgpSDKPrefs.h
PGPsdkPrefGetData	M11	Yes	No	pgpSDKPrefs.h
PGPsdkPrefSetData	M11	Yes	No	pgpSDKPrefs.h
PGPsdkPrefSetFileSpec	M11	Yes	No	pgpSDKPrefs.h
PGPsdkSavePrefs	M11	Yes	No	pgpSDKPrefs.h
PGPCopySymmetricCipherContext	L3, L4	Yes	Yes	pgpSymmetricCipher.h

Table 6: PGP SDK Calls in Cryptographic Module

API Call	Service	CM	CB	Header
PGPFreeSymmetricCipherContext	L3, L4	Yes	Yes	pgpSymmetricCipher.h
PGPGetSymmetricCipherSizes	L3, L4	Yes	Yes	pgpSymmetricCipher.h
PGPInitSymmetricCipher	L3, L4	Yes	Yes	pgpSymmetricCipher.h
PGPNewSymmetricCipherContext	L3, L4	Yes	Yes	pgpSymmetricCipher.h
PGPSymmetricCipherDecrypt	L4	Yes	Yes	pgpSymmetricCipher.h
PGPSymmetricCipherEncrypt	L3	Yes	Yes	pgpSymmetricCipher.h
PGPWashSymmetricCipher	L3, L4	Yes	Yes	pgpSymmetricCipher.h
PGPWipeSymmetricCipher	L3, L4	Yes	Yes	pgpSymmetricCipher.h
PGPContextGetRandomBytes	R1	Yes	Yes	pgpUtilities.h
PGPFreeContext	M4	Yes	Yes	pgpUtilities.h
PGPGetContextMemoryMgr	M5	Yes	No	pgpUtilities.h
PGPGetContextUserValue	M3	Yes	No	pgpUtilities.h
PGPGetPGPTimeFromStdTime	M10	Yes	No	pgpUtilities.h
PGPGetSDKErrorState	M8	Yes	Yes	pgpUtilities.h
PGPGetTime	M10	Yes	No	pgpUtilities.h
PGPNewContext	M3	Yes	Yes	pgpUtilities.h
PGPNewContextCustom	M3	Yes	Yes	pgpUtilities.h
PGPResetSDKErrorState	M8	Yes	Yes	pgpUtilities.h
PGPsdkCleanup	M2	Yes	Yes	pgpUtilities.h
PGPsdkInit	M1	Yes	Yes	pgpUtilities.h
PGPSetContextUserValue	M3	Yes	No	pgpUtilities.h
PGPRunSDKSelfTest	M9	Yes	Yes	pgpUtilities.h
PGPRunAllSDKSelfTests	M9	Yes	Yes	pgpUtilities.h

Glossary

API: Application Programming Interface.

Asymmetric Key: a public or private key.

Context: a reference value used to accept (or refer to) an internal PGP SDK state for an ongoing operation. Examples of contexts include “asymmetric cipher context” or “hash context.”

CAPI: Cryptographic API.

Cipher: a cryptographic algorithm used for encryption and decryption.

DAC: Discretionary Access Control; a form of access control provided by certain computer operating systems.

FIPS: Federal Information Processing Standards.

FIPS 140-1: FIPS for cryptographic modules.

FIPS Mode: FIPS 140-1 compliant mode of operation for PGP SDK.

High-level cryptographic: a high-level CAPI that abstracts away the details of the cryptographic algorithms to be used.

Key Pair: a pair of public / private asymmetric keys.

Key Set: a collection of asymmetric keys.

Low-level cryptographic: a low-level CAPI that includes the intimate details for specific cryptographic algorithms.

MAC: Message Authentication Code.

NIST: National Institute of Standards and Technology.

OpenPGP Message Format: the message-exchange packet formats used by OpenPGP and all PGP products. See "OpenPGP Message Format," draft-ietf-openpgp-formats-07.txt (work in progress).

Option List: a list of options that indicates how processing should proceed.

PGP: Pretty Good Privacy; an application and protocol (RFC 1991) for secure e-mail and file encryption developed by Phil R. Zimmermann. Originally published as Freeware, the source code has always been available for public scrutiny. PGP uses a variety of algorithms, like IDEA*, RSA, DSA, MD5, SHA-1 for providing encryption, authentication, message integrity, and key management. PGP is based on the Web-of-Trust model and has worldwide deployment.

PGP SDK: PGP Software Developer's Kit.

Passphrase: a value used for wrapping/unwrapping a key, either an ASCII passphrase (a sequence of ASCII characters) or a binary passphrase (binary data).

Private Key: the secret portion of an asymmetric key pair.

Public Key: the public portion of an asymmetric key pair.

Random Number: a number generated randomly.

Random Pool: a collection of random bytes, global to the PGP SDK.

Signature: an encrypted hash of data that provides authentication and integrity for the data.

Symmetric Key: key material used for symmetric ciphers. A symmetric key can be provided by the operator or created as needed by the PGP SDK.

User ID: an identifier that is associated with an asymmetric key (via a Signature) that represents the entity (e.g., user) to which the key is assigned.