GuardianEdge
TECHNOLOGIES

Encryption Plus® Cryptographic Library Security Policy

Encryption Plus Cryptographic Library Version 1.0.5
Document Version 2010032
Rama Vissapragada

Symantec Corporation.
350 Ellis St, PO Box 7011• Mountain View, CA 94043 USA
Phone:(650) 527-0217• Fax: (650) 527-1984
http://www.symantec.com

Introduction

This is the non-proprietary security policy for the Encryption Plus® Cryptographic Library ("EPCL"), which provides cryptographic services for GuardianEdge Data Protection Framework, GuardianEdge Hard Disk Encryption, GuardianEdge Encrypted Drive Manager, Guardian Edge Drive Encryption Client, GuardianEdge Removable Storage Encryption, Encryption Anywhere Hard Disk, Encryption Anywhere Removable Storage, Encryption Anywhere CD-DVD, Encryption Plus Hard Disk, Encryption Plus Email, and Encryption Plus Folders.

This security policy fulfills the requirements given in Federal Information Processing Standards Publication 140-2 (FIPS 140-2) Appendix C, as published by the National Institute of Standards and Technology (NIST) of the United States Department of Commerce.

The EPCL is a compact and fast encryption library that provides an Application Programming Interface (API) featuring NIST-approved

- AES encryption,
- SHA-1 hashing,
- HMAC-SHA-1 hashing,
- SHA-2 hashing and
- Pseudo random number generation.

This document outlines the functionality provided by the EPCL and provides high-level details on the means by which the EPCL satisfies FIPS 140-2 requirements. It describes the various services offered by the EPCL library and the mechanisms provided to ensure that these services meet the FIPS 140-2 level 1 requirements.

This security policy is one part of the FIPS 140-2 submission package, which contains additional vendor evidence and source code listings. The entire EPCL submission package is copyright 2004-2010 GuardianEdge Technologies Inc. This document, however, may be freely distributed in an unmodified form.

For more information about the Encryption Plus and Encryption Anywhere lines of products, please visit www.guardianedge.com

For more information on NIST and the Cryptographic Module Validation Program, please visit http://www.nist.gov/cmvp/

1. EPCL Overview

The EPCL provides cryptographic services to the Symantec, GuardianEdge, Encryption Anywhere, and Encryption Plus families of computer security products.

The following operating system platforms were used to operationally test and validate the EPCL to FIPS 140-2 Level 1 requirements. For the purposes of FIPS 140-2 validation[1], the library is provided as a 32-bit and 64-bit dynamically linked library (DLL) that runs on Microsoft Windows operating systems (Windows) or as a (SO) that runs on Apple Computer Mac operating systems (Mac). The EPCL is supported and tested on:

Operating System configurations:

Operating System	Hardware Platform	Processors
Microsoft Windows 7 (x32 bit and x64 bit)	Dell Precision 490 Lenovo x301	Intel Core 2 Duo
Microsoft Windows Vista (x32 bit and x64 bit)	Dell Precision 490	Intel Core 2 Duo
Microsoft Windows XP (x32 bit and x64 bit)	Lenovo x301 Dell Precision 490	Intel Core 2 Duo
Microsoft Windows Server 2008 (x32 bit and x64 bit)	Dell Precision 490	Intel Core 2 Duo
Apple Computer Mac OS X (x32 bit and x64 bit)	Mac mini Macmini3.1	Intel Core 2 Duo

FIPS 140-2 Tested Platforms

EPCL is a software-only module that runs on a multi-chip standalone device. The software library is contained in a file in the form of a,

- EPCL32.DLL, EPCL.DLL, or EPCL.SO - Shared Library

It is intended to meet the requirements of FIPS 140-2 security level 1. For the purposes of this document, the DLL version of the EPCL is referred to as the “EPCL”, the “library”, or the “module”. The operational test platform for the purposes of FIPS 140-2 validation was a General Purpose Computer.

The EPCL additionally supports/runs on the following operating systems, without recompilation:

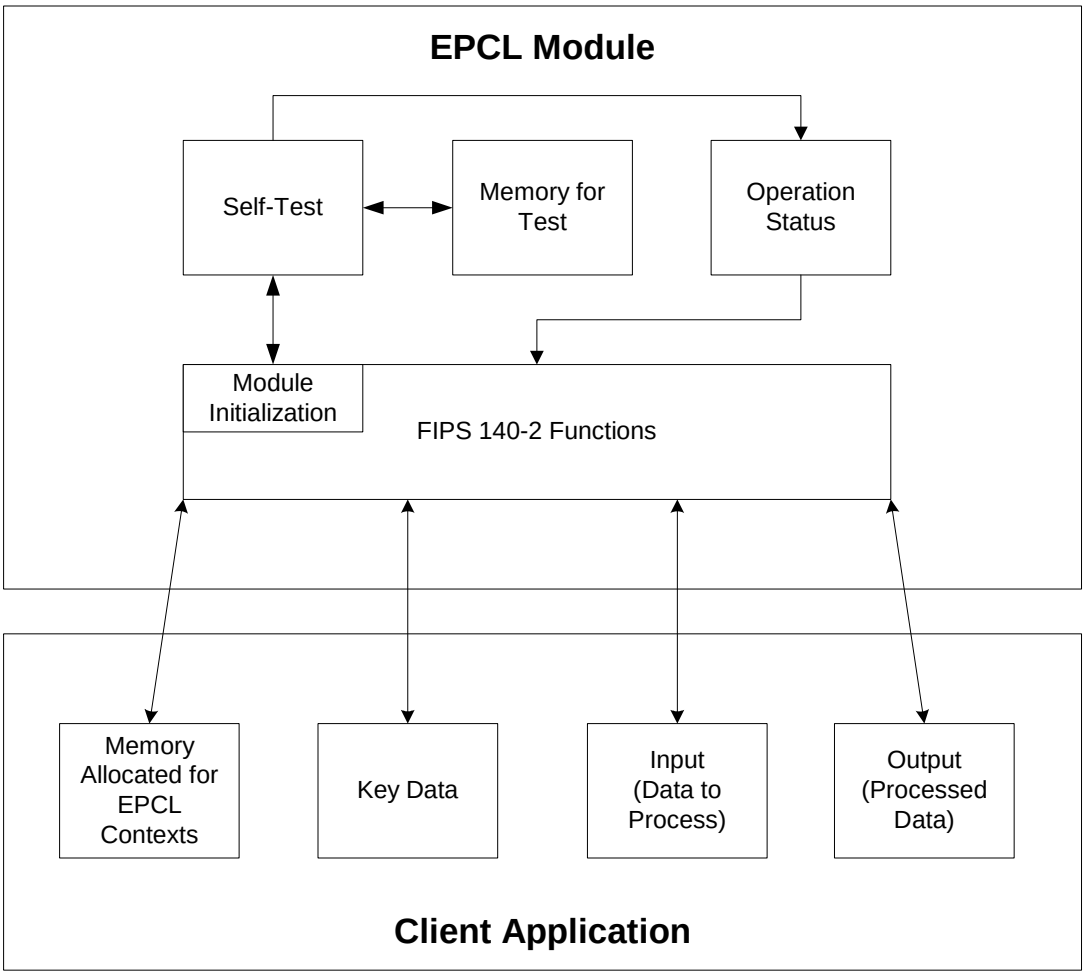
- NT 4.0
- Red Hat Enterprise Linux
- Microsoft Windows 2000

[1] Revalidation of FIPS 140-2 Level 1 Cert #515

Cryptographic Boundary

The physical cryptographic boundary for the EPCL is defined as the enclosure of the computer on which the cryptographic module is installed. The logical boundary is defined as the EPCL32.DLL library.

The block diagram below shows the cryptographic boundary and the relationship between the EPCL and the client applications that attach to it.



Physical Security As a software only product, the FIPS 140-2 physical security requirements are not applicable.

Single Operator Only

The EPCL only supports a single operator. Multiple concurrent operators are not supported.

Approved Algorithms

The EPCL implements only FIPS-approved and FIPS-validated cryptographic algorithms. They are shown in the following table:

Service Type	**Algorithm**	**FIPS**	**Key Size(s) in Bits**	**Modes**	**Certificate Number**
Symmetric Cipher	AES	197	128, 256	CBC, ECB	1420
Message Digest	SHA-1	180-1	N/A	N/A	1288

Message Authentication	HMAC-SHA-1	198	512	N/A	838
Random Number Generation	RNG	186-2, App. 3.1	N/A	N/A	777
Message Digest	SHA-2	180-2	N/A	N/A	1288

The EPCL does not operate in non-FIPS-approved mode.

2. Roles

EPCL implements the Cryptographic Officer (CO) and User roles. There is no maintenance role or maintenance interface. The EPCL does not provide any identification or authentication of its own.

The User is an entity that can access services provided by the EPCL. The User role is implicitly selected when a process calls an API function that is allocated to the User role in the EPCL.

CO is any entity that can install the EPCL onto the computer system, configure the operating system, or access CO services provided by the EPCL. When an entity calls a CO function, it assumes the Cryptographic Officer role. The Cryptographic Officer's role is implicitly selected when installing the EPCL, configuring the operating system, or calling a CO function.

Installation is accomplished by running an installation program. The Cryptographic Officer must have permission to write the DLL constituting the EPCL into the Windows system directory; typically, this requires administrator access to the operating system.

3. Services

EPCL's services consist entirely of an Application Programming Interface (API), which is a collection of functions that can be called from an operating system process. All services provided by the EPCL are described in this section. The module API provides the logical interfaces for data input, data output, control input, and status output.

All functions exported from the EPCL have names starting with the string "Epcl". The functions are implemented using a C language interface, in order to make them available to the greatest number of programming languages.

Summary of Cryptographic Functions

Function	Purpose
EpclAesSetup	Sets up the use of the AES cipher with the given key and other properties
EpclRelease	Can release any sort of context that the library can produce. This corresponds to the FIPS 140-2 required “Zeroization” service.
EpclEcbSetup	Sets up the use of the ECB mode for AES
EpclCbcSetup	Sets up the use of the CBC mode with for AES
EpclEncrypt	Encrypts data using specified cipher and mode
EpclDecrypt	Decrypts data using specified cipher and mode
EpclAesInit	Brings into a client the data for the AES cipher
EpclAesEcbInit	Brings into a client the data for AES symmetric encryption in ECB cipher mode
EpclAesCbcInit	Brings into a client the data for AES symmetric encryption in CBC cipher mode
EpclSha1Init	Brings into a client the data for the SHA-1 hash
EpclSha1Setup	Sets up a SHA-1 context
EpclHashAdd	Passes the message or part of the message for hash (SHA-1) processing
EpclHashOut	Finalizes hash processing and produces an output message hash
EpclSha1HmacInit	Brings into a client the code and data for working with HMAC and for testing HMAC with SHA-1
EpclHmacSetup	Sets up an HMAC context
EpclHmacAdd	Passes the message or part of the message for HMAC processing
EpclHmacOut	Finalizes HMAC processing and produces an output message authentication code
EpclRngSetup	Sets up an RNG context
EpclRngOut	Gets a random pattern
EpclRngInit	Brings into a client the data for the random number generator
EpclGetState	Returns an indicator of the library's overall state. This corresponds to the FIPS 140-2 required “Show Status” service.
EpclGetVersion	Returns the EPCL interface version
EpclTest	Tests all library features that have been initialized
EpclInit	Initializes all library features
EpclSha256Init	Brings into a client the data for the SHA-2 hash
EpclSha256Setup	Sets up a SHA-2 has context

Note that the FIPS 140-2 required self-test on demand service can be performed by reloading the library into memory.

4. Key Management

EPCL performs limited key management. Because the EPCL is a DLL, each process requesting access is provided its own instance of the EPCL. The EPCL contains only keys or data placed into the EPCL via the services described in this document. No keys or data are persistently maintained by the EPCL, or maintained after a process detaches from the EPCL.

Key Generation

The EPCL does not provide key generation services. Keys are generated outside the module. All keys must be entered by the operator; keys are electronically entered, but manually established.

Key Material and Key Storage

The EPCL does not provide any persistent storage of key material. Keys are entered by the operator only via API calls. Key material is stored in the context, which is maintained in a user-supplied data structure passed in each API call. No key material is maintained inside the EPCL between API calls. The only key material used by the EPCL outside of the user-supplied context is that which is stored temporarily in local variables on the stack.

The EPCL relies upon the operating system memory protection to prevent processes from accessing each other's key material. To ensure that other processes cannot access keys and data, the caller must not utilize shared memory. In addition, the operating system page file must not be configured to reside on a network drive.

Key Zeroization

The user must call `EpclRelease` and power cycle the module when finished, in order to zero a context.

Pseudo-Random Number Generator Seeds

As noted above, key generation occurs *outside* of the module. A calling application can request the EPCL's pseudo-random number generator (RNG) whenever it requires random data. If the RNG is used by a calling application in order to perform key generation or other cryptographic operations outside the module, it is important that the generator be seeded with unpredictable values.

List of Critical Security Parameters

- XKEY — value input into the module to key the FIPS 186-2 RNG
- XSEED — value input into the module to seed the FIPS 186-2 RNG
- HMAC-SHA-1 Keys — key size is 512 bits (64 bytes)
- AES Keys

5. Self-Tests

As required by FIPS 140-2, EPCL automatically performs power-on self-tests and continuous self-tests during operation.

Power-On Self-Tests

For this EPCL, “power-on” is when a process attempts to attach to the EPCL. At this time, EPCL performs the following types of tests.

Known Answer Tests

EPCL performs known answer tests for AES, SHA-1, SHA256, HMAC-SHA-1, and FIPS 186-2 random number generation. If the computed result differs from the expected result that is hard-coded into the EPCL, the test fails. The RNG known answer test is performed by entering a fixed seed into the RNG and comparing the resulting pseudo-random number to a known answer.

Software Integrity Test

EPCL computes a 160-bit HMAC-SHA-1 hash of the DLL or SO and compares it to an embedded hash that was placed into the end of the DLL or SO when the EPCL was produced. If the computed hash differs from the hash that is hard-coded into the EPCL, the test fails.

If any of these tests fails, EPCL returns to the error state, and refuses to allow the process to attach to the DLL or SO. Thus, the cryptographic services are not available.

Conditional Self-Tests

During operation, EPCL performs one type of continuous test. Whenever a block of pseudo-random patterns is generated as a result of a call to `EpclRngOut`, the block is compared to the previous block. If the two match, the `EpclRngOut` returns an error.

6. Security Rules

1. The EPCL shall only provide NIST-approved cryptographic algorithms.
2. The EPCL shall support two roles: the Cryptographic Officer and the User role.
3. The EPCL shall not require human intervention to perform power-on self tests.
4. The EPCL shall not support a bypass capability.
5. The EPCL shall not support manual key entry.
6. The EPCL shall not support a maintenance role or maintenance interface.
7. The EPCL shall inhibit all data output during self-tests, zeroization, and error states.

7. Mitigation of Specific Attacks

EPCL is not designed to mitigate specific attacks outside the scope of FIPS 140-2 requirements.

8. Tabular Summaries

Table 1 - EPCL Security Level Specification

The cryptographic module meets the overall requirements applicable to Level 1 security of FIPS 140-2.

Security Requirements Section	Level
Cryptographic Module Specification	1
Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

Table 2 - Roles and Required Identification and Authentication

Role	Type of Authentication	Authentication Data
User	None	N/A
Crypto Officer	None	N/A

Table 3 - Strengths of Authentication Mechanisms

Authentication Mechanism	Strength of Mechanism
User	None
Crypto Officer	None

Table 4 - Services Authorized for Roles

Role	Authorized Services
User	EpclEncrypt EpclDecrypt EpclGetState
Crypto Officer	EpclAesSetup EpclRelease

Role	Authorized Services
	EpclEcbSetup
	EpclCbcSetup
	EpclAesInit
	EpclAesEcbInit
	EpclAesCbcInit
	EpclSha1Init
	EpclSha1Setup
	EpclHashAdd
	EpclHashOut
	EpclSha1HmacInit
	EpclHmacSetup
	EpclHmacAdd
	EpclHmacOut
	EpclRngSetup
	EpclRngOut
	EpclRngInit
	EpclGetVersion
	EpclTest
	EpclInit
	EpclSha256Init
	EpclSha256Setup

Table 5 - Access Rights within Services

Service	CSP	Type of access
EpclAesSetup	AES keys	Write
EpclRelease	AES keys, HMAC keys, XKEY or None	Write to zeroize
EpclEcbSetup	AES keys	Read
EpclCbcSetup	AES keys	Read
EpclEncrypt	AES keys	Execute
EpclDecrypt	AES keys	Execute
EpclAesInit	None	N/A
EpclAesEcbInit	None	N/A

Service	CSP	Type of access
EpclAesCbcInit	None	N/A
EpclSha1Init	None	N/A
EpclSha1Setup	None	N/A
EpclHashAdd	None	N/A
EpclHashOut	None	N/A
EpclSha1HmacInit	None	N/A
EpclHmacSetup	HMAC keys	Write
EpclHmacAdd	HMAC keys	Execute
EpclHmacOut	HMAC keys	Execute
EpclRngSetup	XKEY	Write
EpclRngOut	XSEED	Write and Execute
EpclRngInit	None	N/A
EpclGetState	None	N/A
EpclGetVersion	None	N/A
EpclTest	None	N/A
EpclInit	None	N/A
EpclSha256Init	None	N/A
EpclSha256Setup	None	N/A

9. Acronyms

API – Application Programming Interface

CO – Cryptographic Officer

DLL – Dynamically Linked Library

EPCL – Encryption Plus® Cryptographic Library

RNG – Random Number Generator

SHA-2 – Secure Hashing Algorithm-256 (Message Digest Size 256 bits)

SO – Shared Object