XYPRO XYGATE® /ESDK
by XYPRO
Version 2.0.0

FIPS 140-2 Non-Proprietary Security Policy
Revision 1.22

Level 1 Validation

November 2007

Table of Contents

Introduction

The sections that follow introduce this document, including the purpose for it, references used and the organizational structure of it.

Purpose

This is a non-proprietary Cryptographic Module Security Policy for the XYGATE /ESDK (Encryption Software Development Kit) by XYPRO. This security policy describes how the XYGATE /ESDK by XYPRO meets the security requirements of FIPS 140-2 and how to run the module in a FIPS 140-2 approved mode of operation. This policy was prepared as part of the Level 1 FIPS 140-2 validation of the module.

FIPS 140-2 (Federal Information Processing Standards Publication 140-2 — *Security Requirements for Cryptographic Modules*) details the U.S. Government requirements for cryptographic modules. More information about the FIPS 140-2 standard and validation program is available on the National Institute of Standards and Technology (NIST) website at http://csrc.nist.gov/cryptval/.

References

This document deals only with operations and capabilities of the module in the technical terms of a FIPS 140-2 cryptographic module security policy. More information is available on the module from the following sources:

- The XYPRO website (www.xypro.com) contains information on the full line of products from XYPRO.
- The NIST Validated Modules website (http://csrc.ncsl.nist.gov/cryptval/) contains contact information for answers to technical or sales-related questions for the module.
- The Accredited Standards Committee X9 website (http://www.x9.org/) contains information regarding the American National Standards Institute (ANSI) X9.31 random number generator (RNG).
- RSA Labs website (http://www.rsa.com/rsalabs/) contains information regarding the various Public Key Cryptography Standards.

Document Organization

The Security Policy document is one document in a FIPS 140-2 Submission Package. In addition to this document, the Submission Package contains:

- Vendor Evidence document
- Finite State Machine
- Other supporting documentation as additional references

This Security Policy and the other validation submission documentation were produced by Corsec Security, Inc. under contract to XYPRO. With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Validation Documentation is proprietary to XYPRO and is releasable only under appropriate non-disclosure agreements. For access to these documents, please contact XYPRO.

XYGATE® /ESDK BY XYPRO

Overview

The XYGATE /ESDK library is packaged as a dynamically linked library. The library provides the following basic functionalities:

- Symmetric key encryption including Advanced Encryption Standard (AES), Triple Digital Encryption Standard (Triple-DES), Skipjack, Digital Encryption Standard (DES) and others
- Hashing algorithms including Secure Hash Algorithm-1 (SHA-1), SHA-256, Message Digest 5 (MD5), Hashed Message Authentication Code (HMAC) and others
- Public key encryption including Rivest-Shamir-Adleman (RSA)
- Signature algorithms including RSA, Digital Signature Algorithm (DSA), ElGamal, and others
- Secure session protocols such as Secure Shell (SSH), Secure Sockets Layer (SSL), and Transport Layer Security (TLS)
- E-mail protocols such as Pretty-Good-Privacy (PGP) and Secure Multipurpose Internet Mail Extensions (S/MIME)

The *Cryptographic Key Management* section below details which of the above-referenced algorithms are (and are not) approved for use in a FIPS mode of operation.

Module Interfaces

The XYGATE /ESDK by XYPRO (Version 2.0.0) is classified as a Multi-chip standalone module for FIPS 140-2 purposes. The module has been tested on the following platforms:

- Hewlett-Packard (HP) Nonstop Server G06.29 (ESDKLIB)
- HP Nonstop Server H06.07 (ESDKLIB)
- HP-UX 11.11 (esdklib.sl)
- Solaris 10 (esdklib.so)
- Windows XP w/ SP2 (esdklib.dll)

The logical cryptographic boundary contains the library file (as listed above) and the signature file (esdksig) that contains the HMAC value of the library and the signature file. The physical cryptographic boundary of the XYGATE /ESDK is defined by the metal enclosure around the board. The module supports the

physical interfaces of a general purpose computer (GPC). The physical interfaces include the computer keyboard port, optical drives, floppy disk, mouse, serial ports, parallel ports, networks ports, monitor port, and power plug. The functional module interface exists in the software. See Figure 1 for a standard GPC block diagram.

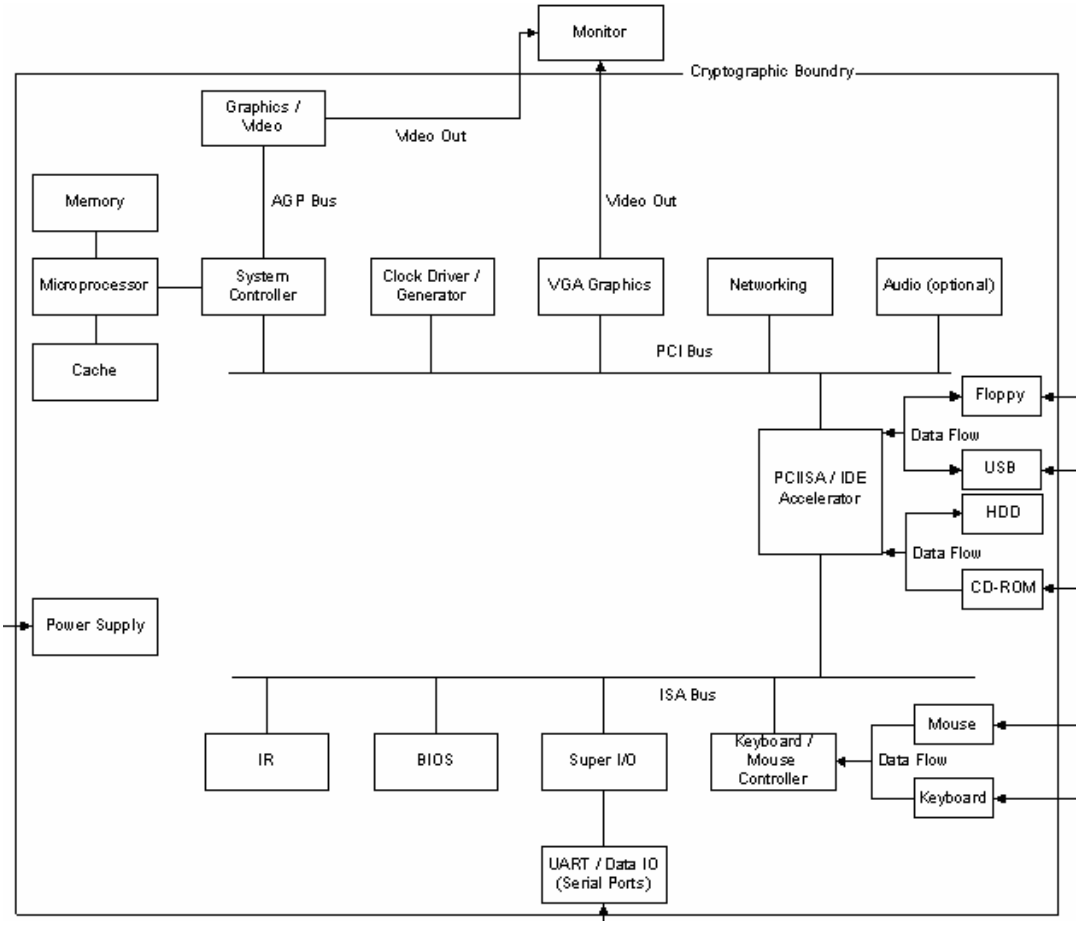

Figure 1 - Standard GPC Block Diagram

The module provides scripts and graphical user interfaces to interact with the components. Figure 2 below shows the module's logical cryptographic boundary.

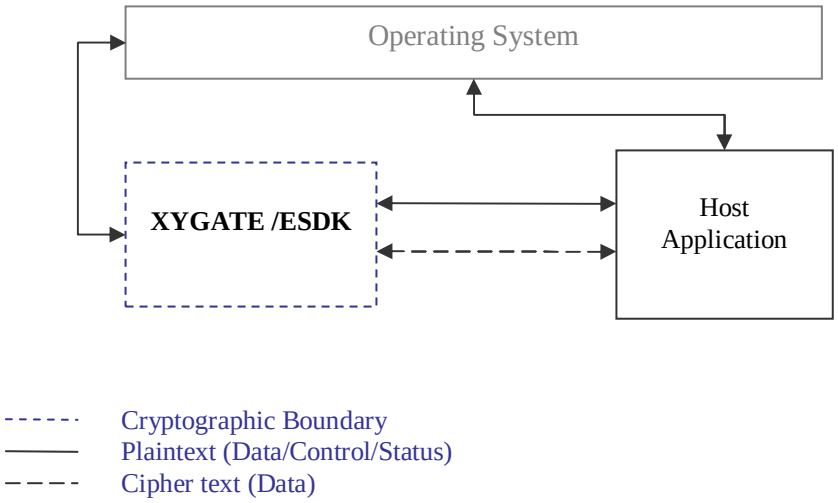

Figure 2 – Logical Cryptographic Boundary

All of these physical interfaces are separated into logical interfaces defined by FIPS 140-2, as described in the following table:

Module Physical Interface	FIPS 140-2 Logical Interface	Module Mapping
PC Network Port, Keyboard Interface, Mouse port	Data Input Interface	The ESDK library is the data input interface. It provides an Application Programming Interface (API) to interact with the module. Any program wishing to utilize the functionality of the module should compile and link as a dynamic library.
PC Network Port	Data Output Interface	The API to the ESDK library is the interface for data output. The data output can be in the form of function input/output variables or return values.
PC Network Port, Serial Port, Keyboard port, Mouse port, PC Power Button	Control Input Interface	The API to the ESDK library provides for control input to the module. This input is in the form of parameters to function calls.
Light-Emitting Diodes (LED), PC Monitor, PC Network Port	Status Output Interface	The appropriate function calls and return values for function calls and log files are the interfaces for status output.
PC Power Interface	Power Interface	Not Applicable.

Table 1 – FIPS 140-2 Logical Interfaces

Roles and Services

The module supports role-based operation. There are two main roles in the module (as required by FIPS 140-2) that operators may assume: a Crypto-Officer role and User role. The Crypto-Officer is responsible for initializing the module and configuring it to run in a FIPS approved mode. After the switch to the FIPS approved mode of operation both the Crypto-Officer and User will be able to utilize the functionality of the module. The table below lists the individual functions contained within the module.

cryptAddCertExtension	*cryptFlushData*	*cryptCreateSession*	*cryptDestroyCert*
cryptDeviceOpen	*cryptCAGetItem*	*cryptImportCert*	*cryptQueryCapability*
cryptAddPrivateKey	*cryptGenerateKey*	*cryptCreateSignature*	*cryptDestroyContext*
cryptDeviceQueryCapability	*cryptCheckCert*	*cryptImportKey*	*cryptQueryObject*
cryptAddPublicKey	*cryptGenerateKeyAsync*	*cryptCreateSignatureEx*	*cryptDestroyEnvelope*
cryptEncrypt	*cryptCheckSignature*	*cryptInit*	*cryptSetAttribute*
cryptAddRandom	*cryptGetAttribute*	*cryptDecrypt*	*cryptDestroyObject*
cryptEnd	*cryptCheckSignatureEx*	*cryptKeysetClose*	*cryptSetAttributeString*
cryptAsyncCancel	*cryptGetAttributeString*	*cryptDeleteAttribute*	*cryptDestroySession*
cryptExportCert	*cryptCreateCert*	*cryptKeysetOpen*	*cryptSignCert*
cryptAsyncQuery	*cryptGetCertExtension*	*cryptDeleteCertExtension*	*cryptDeviceClose*
cryptExportKey	*cryptCreateContext*	*cryptPopData*	*cryptUIDisplayCert*
cryptCAAddItem	*cryptGetPrivateKey*	*cryptDeleteKey*	*cryptDeviceCreateContext*
cryptExportKeyEx	*cryptCreateEnvelope*	*cryptPushData*	*cryptUIGenerateKey*
cryptCACertManagement	*cryptGetPublicKey*		

Table 2 – ESDK Cryptographic Library Functions

The available functions are utilized to provide or perform the cryptographic services. The various services offered by the module are described below. Operators of the module implicitly assume a role based on the services of the module that they are using. Since all services offered by the module can only be used by either the Crypto-Officer or the User (never both) the roles are mutually exclusive. The Critical Security Parameters (CSP) used by each service are listed below followed, in parenthesis, by the level of access each service provides to the listed CSP.

Service	Description	Input	Output	CSP	Role
Installing the module	The Crypto-Officer has the ability to install the module.	Command	Result of Installation	None	Crypto-Officer
Uninstalling/ Removing the module	The Crypto-Officer has the ability to delete the module.	Command	Uninstalled Module	Integrity Check Key (delete)	Crypto-Officer
Configuring in FIPS Mode	The Crypto-Officer has to follow the steps outlined in the Secure Operation section to enable FIPS mode.	Command	Result of FIPS mode set	None	Crypto-Officer
Initialize and un-initialize cryptographic operations	All users can change the state of the module by initializing and un-initializing the module.	Command	Initialized Module	Integrity Check Key (read)	User
Perform encryption and decryption	Every user has the ability to encrypt and decrypt data.	Command, optional Key, Input Data	Status Output, Output Data	Symmetric, Public or (full)	User
Perform hashing and HMAC	Every user has the ability to hash and use HMAC functions.	Command, optional Key, Input Data	Status Output, Output Data	HMAC Key (full)	User
Create and manage certificates	Every user can both create and manage certificates. The Crypto-Officer can manage any certificate, while the Users can only manage their own.	Command	Status Output, Certificate	Certificate (full)	User
Generate random keys and numbers	All users can generate random keys and numbers. This random generation uses a FIPS approved Pseudo-Random Number Generator (PRNG).	Command, Seed, Key	Status Output, Random Number	Seed, Entropy Source (read)	User
Create and manage secure sessions	All users can both create and manage secure sessions such as TLS and SSH.	Command	Status Output, Secure Session Information	Symmetric and Public Keys (full)	User
Create and manage secure e-mail protocols	Every user can both create and manage secure e-mail protocols such as PGP and S/MIME.	Command	Status Output, Output Data	Symmetric and Public Keys (full)	User

Table 3 – Services, Descriptions, Inputs and Outputs

Authentication Mechanisms

The module does not support authentication.

Authentication Mechanism	Strength of the implementation
None	There is no method of authentication specified by the ESDK library or its API. While it is possible that implementations using the ESDK will provide their own authentication, it is impossible to predict the estimated strength of those mechanisms.

Table 4 – Estimated Strength of Authentication Mechanisms

Cryptographic Key Management

FIPS Approved Algorithms

The module supports the following approved cryptographic algorithms:

Symmetric Key Algorithms

The table below lists all the symmetric-key cryptographic algorithms that the ESDK supports. The following modes of operation are defined: F – Cipher Feedback, OFB – Output Feedback, ECB – Electronic Code Book, CBC – Cipher Block Chaining.

Algorithm	Modes Implemented	Key Sizes	Certificate Number
AES	CBC, ECB, CFB, OFB	128, 192, 256 bits	505
Triple-DES	CBC, ECB, CFB, OFB	112, 168 bits	515

Hashing Algorithms

Algorithm	Certificate Number
SHA-1	576
SHA-256	576
HMAC SHA-1	258

Public Key and Signature Algorithms

Algorithm	Key Sizes	Certificate Number
RSA	1024, 2048, 4096 bits	220
DSA	1024 bits	209

Random Number Generators

Algorithm	Certificate Number
ANSI X9.31 PRNG	284

Non-FIPS Approved Algorithm Allowed in FIPS mode

The module supports the following algorithms that are not FIPS approved but allowed in a FIPS mode of operation.

Public Key Algorithms

Algorithm
Diffie-Hellman (provides 80 to 152 bits of encryption strength)
RSA key wrapping (provides 80 to 152 bits of encryption strength)

Non-FIPS Approved Algorithms

The module supports the following non-approved cryptographic algorithms:

Symmetric Key Algorithms

Algorithm	Modes Implemented	Key Sizes
Blowfish	CBC, ECB, CFB, OFB	442 bits
CAST-128	CBC, ECB, CFB, OFB	128 bits
Digital Encryption Standard (DES) (non-compliant)	CBC, ECB, CFB, OFB	40 – 56 bits
International Data Encryption Algorithm (IDEA)	CBC, ECB, CFB, OFB	128 bits
Rivest Cipher 2 (RC2)	CBC, ECB, CFB, OFB	40-1024 bits
RC4	CBC, ECB, CFB, OFB	40-2048 bits
RC5	CBC, ECB, CFB, OFB	40-832 bits
Skipjack (non-compliant)	CBC, ECB, CFB, OFB	80 bits

Hashing Algorithms

Algorithm
Message Digest 2 (MD2)
MD4
MD5
RACE Integrity Primitives Evaluation Message Digest-Message Digest-160 (RIPE-MD)
HMAC MD5
HMAC RIPE-MD

Public Key Algorithms

Algorithm
ElGamal
Diffie-Hellman (non-compliant less than 80-bits provided down to 56-bits)
RSA key wrapping (non-compliant less than 80-bits provided down to 56-bits)
512, 768-bit RSA

Random Number Generators

Algorithm
SHA-1 PRNG

The module supports the following critical security parameters:

Cryptographic Key	Description	Key Type	Storage and Zeroization
Private/public keys	These are keys used in Certificates or in public key encryption schemes in the ESDK.	These keys are either RSA or DSA.	Public and private keys are stored within cryptographic contexts, which are resident in page-locked memory (if available). The keys are explicitly zeroized and deleted before the context that contains the keys is deleted.
Symmetric keys	These are keys used in the symmetric key algorithms of the ESDK.	These keys are AES or Triple-DES.	Symmetric keys follow the same rules for storage and zeroization as public and private keys.
HMAC keys	These keys are used to generate MAC values to authenticate data.	These keys are HMAC-SHA-1	HMAC keys follow the same rules for storage and zeroization as other keys.
PRNG Seed	Random entropy data collected from various sources that are used to seed the ANSI X9.31 PRNG.	Random entropy data	The PRNG seed is stored in volatile memory and is zeroized when a new seed is generated.
Integrity Check Key	Externally generated key used to verify integrity of the module.	HMAC-SHA-1	The key is hard-coded into the source code of the module and is deleted when the module is deleted.

Table 5 – Keys and CSPs

Key Generation

The module implements a FIPS-approved Pseudo-Random Number Generator (PRNG) based on the American National Standards Institute (ANSI) X9.31 specifications. It uses this to generate all symmetric keys.

Key Establishment

Public keys can be established via issuance of a public key certificate by a Certificate Authority (CA) external to the module. Alternatively, the ESDK can be used to act as its own CA.

Symmetric keys used are generated by a PRNG. If necessary, they can be exported by using either a public key encryption context, a certificate containing a public key, or a symmetric key generated through the use of a shared secret pass phrase.

Key Storage

Public keys are stored in certificates or within cryptographic contexts. Private keys are also used within cryptographic contexts or can be stored in PKCS #15 encrypted files within the file system. Symmetric keys are only stored within a cryptographic context. Cryptographic contexts are resident in page-locked memory (If the operating system supports it). If not, they are stored in regular memory.

Key Usage

Users have complete access to all keys and CSPs with the exception of the PRNG seed and the Integrity Check Key. Users can create, read, write, update, and destroy (zeroize) all symmetric and HMAC keys used by the module. The PRNG seed is managed by the module and cannot be directly modified in any way, though when a random number is generated a new seed is created. The Integrity Check Key is hard-coded into the module and cannot be accessed by Users. The Crypto-Officer can delete the key by deleting the module. This is done by deleting the directory that contains the module library.

Key Zeroization

All keys are zeroized when the cryptographic context they are a part of is deleted. Contexts can be deleted either explicitly, or implicitly when the cryptEnd() function is called.

Self-Tests

The module performs power-up and conditional self-tests to ensure the secure and proper operation. The sections below provide details on the module's self-tests. See the *Cryptographic Key Management* section above for a description of which algorithms are FIPS-Approved and non-FIPS-Approved.

Power-Up Self-Tests

The power-up self-tests implemented include known answer tests (KAT) for Approved algorithms (Triple-DES, AES, SHA-1, SHA-256, HMAC-SHA-1, and

RSA) and non-Approved algorithms (Blowfish, CAST-128, DES, IDEA, RC2, RC4, RC5, Skipjack, MD2, MD4, MD5, RIPE-MD, HMAC-MD5, HMAC-RIPE-MD, and Diffie-Hellman). Also executed at power-up is a PRNG KAT, pairwise consistency tests for DSA (FIPS-Approved) and ElGamal (non-FIPS-Approved), and a software integrity check with HMAC SHA-1.

Because the module is a software library, the power-up self-tests are run when a host application performs the call to the cryptInit() function. Linking the module invokes certain API function calls, which triggers the self tests. If these self tests fail, then the API calls fail and the API will not be functional.

Conditional Self-Tests

The module performs two conditional self-tests: a pairwise consistency test each time the module generates a DH or ElGamal public/private key and a continuous random number generator test each time the module produces random data. The seed for the PRNG is also tested. If any error occurs during the tests, the cryptographic context that requested the keys or random numbers will be deactivated until a proper key or random number is generated again and passes the self test, or until the context is destroyed and a new one created and another request made.

Design Assurance

XYPRO stores their source code within Microsoft Visual SourceSafe. Code must be checked out to edit and then checked in to become part of the final source tree for the ESDK. Corsec Security, Inc. also stores documents in Visual SourceSafe to ensure that documents are not tampered with and are only edited by those who have the proper permissions.

This Security Policy describes the secure operation of the XYGATE /ESDK module, specifies the procedures for secure installation, initialization, startup, and operation of the module, and provides guidance for use by Crypto-Officers and Users.

Mitigation of Other Attacks

In a FIPS mode of operation, the module does not claim to mitigate any attacks.

SECURE OPERATION

The XYGATE /ESDK module meets Level 1 requirements for FIPS 140-2. The sections below describe how to place and keep the module in a FIPS-approved mode of operation.

Initial Setup

The module has to be installed on one of the tested platforms, which are Windows XP w/ SP2, Solaris 10, HP-UX B 11.11, HP Nonstop G06.29 Server and HP Nonstop H06.07 Server. The platform must be setup for a single-user operation mode. Installing the module is simply ensuring that the ESDK dynamic library is in the same directory as the accompanying application.

A Note On The Installation For HP Nonstop Platforms:

While an initial digital signature for the module is provided by the vendor for the software integrity test, the operating system (OS) modifies the module binary upon the first execution (a result of OS linkage). Thus, the digital signature generated by the module for the integrity test will not match the original digital signature. In order to properly install the module, the following procedures must be followed:

1. Install the module (esdklib).
2. Run the application using esdklib. This will create the first-time-execution link changes to the esdklib.
3. Generate a signature for the modified esdklib and place it in esdksig.

This will ensure the module is properly installed. These installation procedures are required each time the system is rebooted.

Crypto-Officer Guidance

The Crypto-Officer is required to ensure that two functions are called before any other API calls are made. The first is cryptlib_fips_set(1), which ensures the self-tests are performed and that only the FIPS approved algorithms can be used. The second is cryptInit(), which performs the power-up self-tests and integrity tests. All responsibilities for ensuring FIPS mode lie with the Crypto-Officer.

User Guidance

When using key establishment protocols (RSA and DH) in FIPS approved mode, the user is responsible for selecting a key size that provides the appropriate level of key strength for the key being transported.

ACRONYMS

Acronym	Definition
Triple-DES	Triple Digital Encryption Standard
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
API	Application Programming Interface
CA	Certificate Authority
CBC	Cipher Block Chaining
CFB	Cipher Feedback
CSP	Critical Security Parameters
DES	Digital Encryption Standard
DSA	Digital Signature Algorithm
ECB	Electronic Code Book
ESDK	Encryption Software Development Kit
FIPS	Federal Information Processing Standard
GPC	General Purpose Computer
HMAC	Hashed Message Authentication Code
HP	Hewlett-Packard
IDEA	International Data Encryption Algorithm
KAT	Known Answer Test
MD 2,4,5	Message Digest 2,4,5
NIST	National Institute of Standards and Technology
OFB	Output Feedback
OS	Operating System
PC	Personal Computer
PGP	Pretty Good Privacy
PKCS	Public Key Cryptography Standard
PRNG	Pseudo Random Number Generator
RC 2,4,5	Rivest Cipher 2,4,5
RIPE-MD	RACE Integrity Primitives Evaluation Message Digest
RNG	Random Number Generator
RSA	Rivest, Shamir, Adleman
S/MIME	Secure Multipurpose Internet Mail Extensions
SHA	Secure Hash Algorithm
XESDK	XYGATE Encryption Software Development Kit