The Security Division of EMC

RSA BSAFE® Crypto-Kernel 1.3.1 Security Policy

This is a non-proprietary Security Policy for the RSA BSAFE Crypto-Kernel 1.3.1 (Crypto-Kernel) software toolkit. This document describes how Crypto-Kernel meets the security requirements of FIPS 140-2, and how to securely operate Crypto-Kernel in a FIPS 140-2-compliant manner. This policy is prepared as part of the FIPS 140-2 Level 1 validation of Crypto-Kernel.

FIPS 140-2 (Federal Information Processing Standards Publication 140-2 - Security Requirements for Cryptographic Modules) details the United States Government requirements for cryptographic modules. For more information about the FIPS 140-2 standard and validation program, see the NIST Web site at `http://csrc.nist.gov/cryptval/`.

Contents:

1 Introduction

This document deals only with the operations and capabilities of Crypto-Kernel in the technical terms of a FIPS 140-2 cryptographic module security policy. More information about Crypto-Kernel and the entire RSA BSAFE product line is available from the following resources:

- Information on the full line of RSA products and services is available at `http://www.rsa.com/`.
- RSA BSAFE product overviews are available at `http://www.rsa.com/node.asp?id=1204`.
- Answers to technical or sales related questions are available at `http://www.rsa.com./node.asp?id=1067`.

1.1 Document Organization

This document explains the Crypto-Kernel FIPS 140-2 relevant features and functionality. This document comprises the following sections:

- This section, “Introduction”, provides an overview and introduction to the Security Policy.
- “Crypto-Kernel Module” on page 3 describes Crypto-Kernel and how it meets FIPS 140-2 requirements.
- “Secure Operation of Crypto-Kernel” on page 8 specifically addresses the required configuration for the FIPS 140-2 mode of operation.
- “Services” on page 10 lists all of the functions of Crypto-Kernel.
- “Acronyms and Definitions” on page 12 lists the acronyms and definitions used in this document.
- “Contacting RSA” on page 14 lists the ways to obtain more information about RSA products and services, access support, or provide feedback on the documentation.

2 Crypto-Kernel Module

Crypto-Kernel is a software development toolkit that offers optimized cryptographic algorithm implementations. Crypto-Kernel provides the cryptographic foundation for RSA BSAFE security products designed for C/C++ developers. Crypto Kernel is designed to offer the lowest level cryptographic application programming interfaces.

The features of this release of the Crypto-Kernel include:

- Support for the following cryptographic algorithms:
  - AES (128, 192, and 256-bit key sizes) in ECB, CBC, and XTS modes.

    Note: The AES algorithm in XTS mode is not FIPS 140-2-approved.
  - SHA-1 and SHA-256.
  - HMAC-SHA-1 and HMAC-SHA-256.
- FIPS 140-2-validated cryptography.
- The binaries produced for this release of the Crypto-Kernel are kernel modules targeted at systems running a Windows operating system.

2.1 Cryptographic Module

Crypto-Kernel is classified as a multi-chip standalone cryptographic module for the purposes of FIPS 140-2. As such, Crypto-Kernel must be tested on a specific operating system and computer platform. The cryptographic boundary includes Crypto-Kernel running on selected platforms running selected operating systems while configured in "single user" mode. Crypto-Kernel was validated as meeting all FIPS 140-2 Level 1 security requirements, including cryptographic key management and operating system requirements. Crypto-Kernel is packaged as a kernel module that contains the module's entire executable code. The Crypto-Kernel toolkit relies on the physical security provided by the host PC in which it runs.

For FIPS 140-2 validation, Crypto-Kernel is tested on Microsoft® Windows Server 2003 SP2 on:

- x86 (32-bit)
- x86_64 (64-bit)
- Itanium2 (64-bit)

Compliance is maintained on the following platforms for which the binary executable remains unchanged.

- Microsoft Windows XP Professional SP2 on x86 (32-bit)
- Microsoft Vista Enterprise on x86 (32-bit).

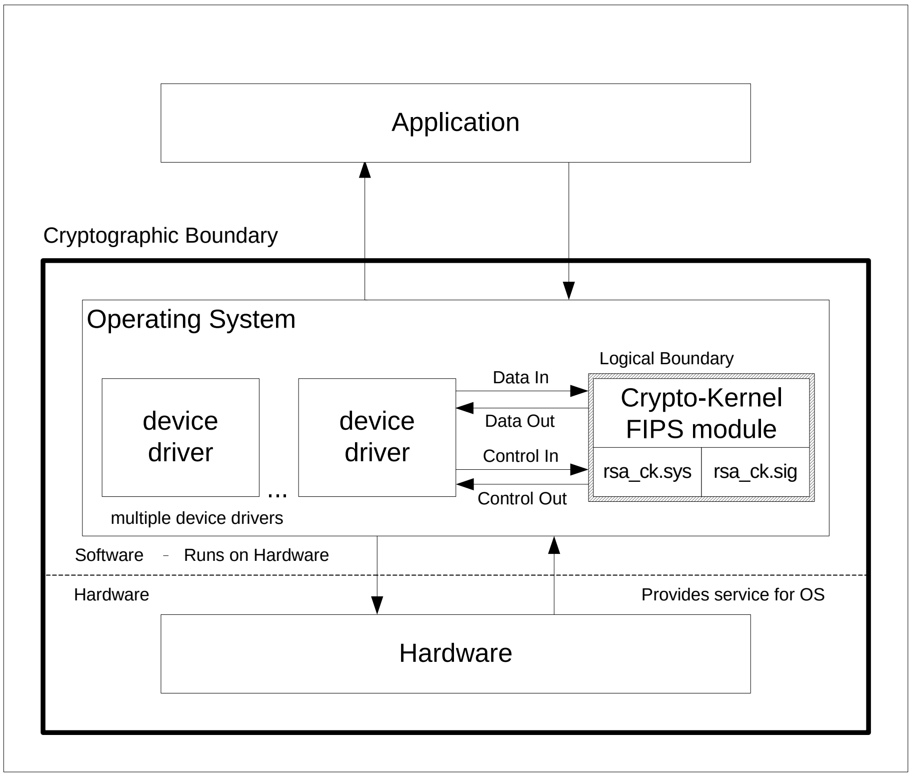
For resolution of the “Multi User” modes issue, see the NIST document, Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program, located at http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf.

2.2 Crypto-Kernel Interfaces

Crypto-Kernel is evaluated as a multi-chip, standalone module. The physical interfaces for Crypto-Kernel consist of the keyboard, mouse, monitor, CD-ROM drive, floppy drive, serial ports, USB ports, COM ports, and network adapter(s).

The logical boundary of the cryptographic module encompasses the files rsa_ck.sys and rsa_ck.sig. The underlying logical interface to Crypto-Kernel is the application programming interface (API), which is documented in the RSA BSAFE Crypto-Kernel 1.3.1 Developer's Guide. Crypto-Kernel provides for Control Input through the API calls. Data Input and Output are provided in the variables passed with the API calls, and Status Output is provided through the returns and error codes that are documented for each call. The following diagram illustrates the Crypto-Kernel logical interfaces.

Figure 1 Crypto-Kernel Logical Interfaces



2.3 Roles and Services

Crypto-Kernel meets all FIPS 140-2 Level 1 requirements for roles and services, implementing both a User role and Officer role. As allowed by FIPS 140-2 Level 1, Crypto-Kernel does not support user identification or authentication for these roles. Only one role can be active at a time and Crypto-Kernel does not allow concurrent operators.

The following table describes the services accessible by the two roles.

Table 1 Crypto-Kernel Roles and Services

Role	Services
User	The User can set the role to Officer, perform cryptographic operations (as described in the *RSA BSAFE Crypto-Kernel Developer's Guide*), and unload the module.
Officer	The Officer can set the role to User, perform cryptographic operations (as described in the *RSA BSAFE Crypto-Kernel Developer's Guide*), unload the module, and invoke the module self-tests on demand.

2.3.1 User Role

An operator assuming the User role can call any Crypto-Kernel function except for `CK_FIPS_run_self_tests()`, which is reserved for the Officer. For the complete list of Crypto-Kernel functions, see "Services" on page 10.

2.3.2 Officer Role

An operator assuming the Officer role can call any Crypto-Kernel function. For the complete list of Crypto-Kernel functions, see "Services" on page 10.

2.4 Cryptographic Key Management

Cryptographic key management is concerned with generating and storing keys, managing access to keys, protecting keys during use, and zeroizing keys when they are no longer required.

2.4.1 Key Generation

Crypto-Kernel does not support key generation in this version of the module.

2.4.2 Key Storage

Crypto-Kernel does not provide long-term cryptographic key storage. Crypto-Kernel stores all cryptographic keys in volatile (short term) memory in plain text.

2.4.3 Key Access

An authorized operator of the module has access to all key data created during Crypto-Kernel operation.

Note: The User and Officer roles have equal and complete access to all keys

The following table lists the services provided by the toolkit with the type of access to keys or CSPs.

Table 2 Key and CSP Access

Service	Key or CSP	Type of Access
Encryption and decryption	AES keys	Read/Execute
Hashing	None	N/A
MAC	HMAC keys	Read/Execute
Self-test (Crypto Officer services)	Hardcoded keys (HMAC)	Read/Execute
Show status	None	N/A
Zeroization	All	Read/Write

2.4.4 Key Protection/Zeroization

Crypto-Kernel accepts key data from calling applications for its cipher and HMAC operations. Secret key data is supplied as a reference to a caller-managed buffer of data. The responsibility of managing this key buffer memory remains with the caller.

Crypto-Kernel uses context data structures to store cipher and digest state information across multiple API calls. A calling application creates the contexts structures and copies key data to the structures. All context data, including key data, is zeroized when the context structures are destroyed at the conclusion of the cipher or digest operations. It is up to the calling application to destroy the context structures after the secure information is no longer required.

The HMAC key used by the integrity check process, and the AES and HMAC Known Answer Test (KAT) keys are compiled into the cryptographic module binary. If these keys are modified the module will fail the integrity check or will fail the KATs and transition to the Module Invalid state.

All keys remain in the process space of a single user. The operating system protects memory and process space from unauthorized access.

For more information, see the steps outlined in the *RSA BSAFE Crypto-Kernel Developer's Guide*.

2.5 Cryptographic Algorithms

FIPS 140-2 requires that FIPS 140-2-approved algorithms are used whenever there is an applicable FIPS 140-2 standard. The following table lists the algorithms supported by Crypto-Kernel, all of which are FIPS 140-2-approved.

Table 3 Crypto-Kernel FIPS 140-2-approved Algorithms

Algorithm	Validation Certificate
AES in ECB and CBC modes (128, 192, and 256-bit)	1105
SHA-1	1028
SHA-256	1028
HMAC-SHA-1	617
HMAC-SHA-256	617

Note: The AES algorithm in XTS mode is not FIPS 140-2-approved.

For more information about using Crypto-Kernel in a FIPS 140-2-compliant manner, see "Secure Operation of Crypto-Kernel" on page 8.

2.6 Module Startup

At startup, Crypto-Kernel loads the `SignaturePath` and `TestMode` registry keys, and performs the module self-tests.

If the registry key loading or self-tests fail, the module transitions to the Module Invalid state. From the Module Invalid state, no actions can be performed except to unload the module.

The module self-tests can be run on demand by an operator assuming the Officer role. For more information, see "Roles" on page 8.

2.6.1 Startup Self-tests

Crypto-Kernel performs the following startup self-tests:

- Module integrity check using HMAC-SHA-256
- AES KATs (key lengths of 128, 192, and 256 bits)
- SHA-1 KAT
- SHA-256 KAT
- HMAC-SHA-1 KAT
- HMAC-SHA-256 KAT.

Startup self-tests are executed automatically when Crypto-Kernel is loaded into memory.

3 Secure Operation of Crypto-Kernel

This section provides an overview of how to securely operate Crypto-Kernel to be in compliance with the FIPS 140-2 standards.

3.1 Crypto User Guidance

In a FIPS 140 mode of operation, a crypto user must only use FIP2 140-2-approved algorithms. As listed in Table 2, "Crypto-Kernel FIPS 140-2-approved Algorithms" on page 9, Crypto-Kernel provides support for a range of FIPS 140-2-approved algorithms. An additional requirement for using the HMAC algorithm is that the key must be between 128 and 4096 bits.

3.2 Roles

The following table lists the roles a user can operate in.

Table 4 Crypto-Kernel Roles

Role Identifier	Description
`CK_FIPS_ROLE_USER`	An operator assuming the User role can call any Crypto-Kernel function except for `CK_FIPS_run_self_tests()`, which is reserved for the Officer. The complete list of Crypto-Kernel functions is outlined in "Services" on page 10.
`CK_FIPS_ROLE_OFFICER`	An operator assuming the Officer role can call any Crypto-Kernel function. The complete list of Crypto-Kernel functions is outlined in "Services" on page 10.

3.3 Modes of Operation

The following table lists and describes the available modes of operation.

Table 5 Crypto-Kernel Modes of Operation

Mode	Description
`CK_FIPS_140_MODE_FIPS` FIPS 140-2-approved.	Enables the FIPS 140-2-approved algorithms listed in Table 3 on page 7. This is the Crypto-Kernel default mode on start up.
`CK_FIPS_140_MODE_FIPS_TEST` Not FIPS 140-2-approved.	A development mode for testing purposes only.

3.4 Operating Crypto-Kernel

Module parameters are supplied to indicate the location of the kernel module binary and the kernel module signature data. The method of supplying the parameters depends upon the platform.

Access to the any of the module functionality, except to retrieve fixed module details, is via a `CK_FIPS` context structure. Applications can create multiple `CK_FIPS` context structures and each `CK_FIPS` context must be used within a single thread of execution. That is, contexts cannot be shared between threads.

Crypto-Kernel can only be changed to `CK_FIPS_140_MODE_FIPS_TEST` mode if the module was initialized properly. A `TestMode` registry entry must be set such that `CK_FIPS_set_mode()` allows changing to `CK_FIPS_140_MODE_FIPS_TEST` mode. Otherwise, an error occurs.

A `CK_FIPS` context is created with a `USER` role and accesses the cryptographic routines in a FIPS approved mode. Both the role and the mode can be changed by the application. Query the current role and mode by calling `CK_FIPS_get_role()` or `CK_FIPS_get_mode()` respectively. Change the `CK_FIPS` context role by calling `CK_FIPS_set_role()`. There is no authentication step required for role changes.

When placing the module in FIPS-approved mode, the module shall be initialized by loading it into memory while the TestMode registry value is set to 0 (or not set). When changing the `CK_FIPS` context mode to the non-Approved mode of operation, the `CK_FIPS_set_mode()` function can be used if the `TestMode` registry key was set to 1 on startup of the module. Crypto-Kernel can only be changed to `CK_FIPS_140_MODE_FIPS_TEST` mode if the module was initialized with the registry key properly set to 1, otherwise an error occurs when `CK_FIPS_set_mode()` is called. `CK_FIPS_set_mode()` cannot be used to change into the approved mode of operation. To be in a FIPS-approved mode of operation, the module shall not be loaded with the `TestMode` registry set to 1.

To avoid unguarded access to global state information, run self tests in a single thread of execution. Run the self test by calling `CK_FIPS_run_self_tests()` using a `CK_FIPS` context that is in `OFFICER` role.

Retrieve product information about the Crypto-Kernel FIPS 140-2 module by calling `CK_FIPS_MODULE_get_info()`. Retrieve validity status information about the module (either the current status or the reason for the current status) by calling `CK_FIPS_MODULE_get_status()`.

Both the `USER` and `OFFICER` role can be used to access the FIPS 140-2-approved cryptographic functions.

Note: Note the following:

- The AES algorithm in XTS mode is not FIPS 140-2-approved and shall not be used in the approved mode of operation.
- Cryptographic keys shall not be shared between modes. For example, a key generated in `CK_FIPS140_MODE_FIPS_TEST` mode must not be shared with an application running in `CK_FIPS_140_MODE_FIPS` mode.

4 Services

The following table lists the functions provided by Crypto-Kernel. For more information about these functions, see the *RSA BSAFE Crypto-Kernel Developer's Guide*.

Table 6 Crypto-Kernel Functions

Function	Description
`CK_FIPS_MODULE_get_info()`	Retrieve product information about the Crypto-Kernel FIPS 140-2 module.
`CK_FIPS_MODULE_get_status()`	Retrieve validity status information about the Crypto-Kernel FIPS 140-2 module.
`CK_FIPS_new()`	Creates a new FIPS management context structure, which is required for calls to all other FIPS management functions. The new context structure is returned in the User role.
`CK_FIPS_free()`	Destroys the FIPS management context structure.
`CK_FIPS_get_mode()`	Returns the current mode of operation.
`CK_FIPS_set_mode()`	Sets a new mode for the operator.
`CK_FIPS_get_role()`	Returns the current role of the operator.
`CK_FIPS_set_role()`	Sets a new role for the operator.
`CK_FIPS_run_self_tests()`	Performs the Crypto-Kernel powerup self-tests. The FIPS management context must be in the Officer role to successfully run the self-tests.
`CK_FIPS_cipher_new()`	Creates a new cipher context structure for encryption and decryption operations according to the FIPS control state held by the FIPS management context.
`CK_FIPS_cipher_free()`	Finalizes the data of the cipher context structure and then frees any allocated memory.
`CK_FIPS_CIPHER_params_set()`	Sets key and initialization vector (IV) data against the cipher context structure.
`CK_FIPS_CIPHER_process()`	Performs the encryption or decryption action.
`R1_CIPH_METH_aes_cbc_fast()`	Returns the method functions for AES encryption in CBC mode that are optimized for performance.

Table 6 Crypto-Kernel Functions (continued)

Function	Description
`R1_CIPH_METH_aes_ecb_fast()`	Returns the method functions for AES encryption in ECB mode that are optimized for performance.
`R1_CIPH_METH_xts_aes_fast()`	Returns the method functions for AES encryption in XTS mode that are optimized for performance. **Note:** Not allowed in FIPS-approved mode.
`CK_FIPS_DIGEST_new()`	Creates a new digest context for either digest or MAC operations. The digest context is dependent upon the FIPS management context.
`CK_FIPS_DIGEST_free()`	Destroys the digest context.
`CK_FIPS_DIGEST_init()`	Initializes the digest operation state before any data is added to it.
`CK_FIPS_DIGEST_update()`	Processes a buffer of input data and updates the state of the digest or MAC operation.
`CK_FIPS_DIGEST_final()`	Completes the digest or MAC operation and returns the digest result.
`CK_FIPS_DIGEST_params_set()`	Sets the key data for a MAC operation.
`CK_FIPS_DIGEST_mac_digest_set()`	Sets the digest method table that is used for a MAC operation.
`CK_FIPS_DIGEST_length()`	Returns the length of the digest.
`R1_DGST_METH_sha256_small()`	Returns the method for the SHA-256 digest.
`R1_DGST_METH_hmac()`	Returns the method for HMAC.

5 Acronyms and Definitions

The following table lists and describes the acronyms and definitions used throughout this FIPS submission documentation.

Table 7 Acronyms and Definitions

Term	Definition
AES	Advanced Encryption Standard. A fast block cipher with a 128-bit block, and keys of lengths 128, 192, and 256 bits. Replaces DES as the US symmetric encryption standard.
API	Application Programming Interface.
Attack	Either a successful or unsuccessful attempt at breaking part or all of a cryptosystem. Various attack types include an algebraic attack, birthday attack, brute force attack, chosen ciphertext attack, chosen plaintext attack, differential cryptanalysis, known plaintext attack, linear cryptanalysis, and middleperson attack.
CBC	Cipher Block Chaining. A mode of encryption in which each ciphertext depends upon all previous ciphertexts. Changing the Initialization Vector (IV) alters the ciphertext produced by successive encryptions of an identical plaintext.
CSP	Critical Security Parameter.
ECB	Electronic Codebook. A mode of encryption that divides a message into blocks and encrypts each block separately.
Encryption	The transformation of plaintext into an apparently less readable form (called ciphertext) through a mathematical process. The ciphertext can be read by anyone who has the key that decrypts (undoes the encryption) the ciphertext.
FIPS	Federal Information Processing Standards.
HMAC	Keyed-Hashing for Message Authentication Code.
IV	Initialization Vector. Used as a seed value for an encryption operation.
KAT	Known Answer Test.
Key	A string of bits used in cryptography, allowing people to encrypt and decrypt data. Can be used to perform other mathematical operations as well. Given a cipher, a key determines the mapping of the plaintext to the ciphertext. The types of keys include distributed key, private key, public key, secret key, session key, shared key, subkey, symmetric key, and weak key.
NIST	National Institute of Standards and Technology. A division of the US Department of Commerce (formerly known as the NBS) which produces security and cryptography-related standards.
OS	Operating System.

Table 7 Acronyms and Definitions (continued)

Term	Definition
PC	Personal Computer.
Privacy	The state or quality of being secluded from the view or presence of others.
SHA-2	The NIST-mandated successor to SHA-1, to complement the Advanced Encryption Standard. It is a family of hash algorithms (SHA-224, SHA-256, SHA-384 and SHA-512) that produce digests of 224, 256, 384 and 512 bits respectively.

6 Contacting RSA

The RSA Web site (www.rsa.com) contains the latest news, security bulletins and information about coming events.

The RSA BSAFE Web site (www.rsa.com/node.asp?id=1204) contains product information.

The RSA Laboratories Web site (www.rsa.com/rsalabs/node.asp?id=2152) contains answers to frequently asked questions.

6.1 Support and Service

If you have any questions or require additional information, see RSA Support (www.rsa.com/node.asp?id=1067) or RSA SecurCare Online (https://knowledge.rsasecurity.com).

6.2 Feedback

We welcome your feedback on the documentation produced by RSA. Please e-mail us at userdocs@rsa.com.