RSA BSAFE®

Crypto-C Micro Edition 1.7.2

FIPS 140-2 Validation Security Policy

Level 1 Validation

RSA SECURITY®

Contact Information

See our Web sites for regional Customer Support telephone and fax numbers.

RSA Security Inc.
www.rsasecurity.com

RSA Security Ireland Limited
www.rsasecurity.ie

Trademarks

License Agreement

Note on Encryption Technologies

This product may contain encryption technology. Many countries prohibit or restrict the use, import or export of encryption technologies and current use, import and export regulations should be followed when exporting this product.

Distribution

RSA Security Notice

The RC5® Block Encryption Algorithm With Data-Dependent Rotations is protected by U.S. Patent #5,724,428 and #5,835,600.

Compaq MultiPrime™ technology is protected by U.S. Patent #5,848,159 and is the subject of patent applications in other countries.

Published April 6, 2004

001-054007-172-005-000

Introduction

This is a non-proprietary RSA Security Cryptographic Module security policy. This security policy describes how the Cryptographic Module meets the security requirements of FIPS 140-2, and how to securely operate the Cryptographic Module in a FIPS-compliant manner. This policy was prepared as part of the level 1 FIPS 140-2 validation of the Cryptographic Module.

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

References

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

- The RSA Security Web site contains information on the company's full line of products and services at http://www.rsasecurity.com/.
- Information on the complete line of RSA BSAFE security SDKs is provided at http://www.rsasecurity.com/products/bsafe/index.html.
- For answers to technical or sales related questions please refer to http://www.rsasecurity.com/contact/.
- Interact with other developers and RSA Security staff to get answers to security and product questions, read security-related articles, and more. Go to RSA Developer Central at http://developer.rsasecurity.com/.

Document Organization

This document explains the Cryptographic Module's FIPS 140-2 relevant features and functionality. This first section, Introduction, provides an overview and introduction to the Security Policy. Cryptographic Module on page 4 describes the Cryptographic Module and how it meets FIPS 140-2 requirements. Secure Operation of the Cryptographic Module on page 11 specifically addresses the required configuration for the FIPS-mode of operation. Services on page 13 lists all of the functions provided by the Cryptographic Module. Acronym List on page 18 lists the definitions for the acronyms used in this document.

With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Validation Submission Documentation is RSA Security-proprietary and is releasable only under appropriate non-disclosure agreements. For access to these documents, please contact RSA Security.

Cryptographic Module

This section provides an overview of the Cryptographic Module. The following topics are discussed:

- Introduction
- Cryptographic Module
- Module Interfaces
- Roles and Services
- Cryptographic Key Management
- Cryptographic Algorithms
- Self-Test.

Introduction

Wireless technology provides easy and fast delivery of information and services through handheld digital devices such as mobile phones, pagers and personal digital assistants (PDAs). Crypto-C Micro Edition can be easily ported to different embedded operating systems and its features include the ability to optimize code for different processors and for specific speed or size requirements.

Crypto-C Micro Edition offers a full set of cryptographic algorithms, including public key operations, symmetric, block and stream ciphers, message digests, message authentication and the Pseudo Random Number Generator (PRNG). Developers can implement the full suite of algorithms through a single Application Programming Interface (API) or select a specific set of algorithms in order to meet performance or resource constraints.

With Crypto-C Micro Edition, companies can easily embed high levels of security and privacy into a wide range of wireless applications without being cryptography experts.

Cryptographic Module

This Cryptographic Module is classified as a multi-chip standalone module for FIPS 140-2 purposes. As such, the module must be tested upon a particular operating system and computer platform. The cryptographic boundary thus includes the Cryptographic Module running on selected platforms running selected operating systems while configured in “single user” mode. The Cryptographic Module running on this platform was validated as meeting all FIPS 140-2 level 1 security requirements, including cryptographic key management and operating system requirements. The Cryptographic Module is packaged as a dynamically loaded module or shared library file which contains all the module’s executable code. Additionally, the RSA BSAFE Crypto-C ME toolkit relies on the physical security provided by the host PC in which it runs.

The RSA BSAFE Crypto-C ME toolkit was tested on the following platforms:

- Red Hat Linux 7.1

  Compliance is maintained on platforms for which the binary executable remains unchanged including (but not limited to):
  - Red Hat Linux 8.0
  - Red Hat Advanced Server 2.1.

- Sun Microsystems Solaris 8 (Sun OS 5.8) Sparc V8+ (32-bit)

  Compliance is maintained on platforms for which the binary executable remains unchanged including (but not limited to):
  - Sun Microsystems Solaris 9 (SunOS 5.9) Sparc V8+.

- Sun Microsystems Solaris 8 (SunOS 5.8) Sparc V9 (64-bit)

  Compliance is maintained on platforms for which the binary executable remains unchanged including (but not limited to):
  - Sun Microsystems Solaris 9 (SunOS 5.9) Sparc V9.

- Microsoft Windows Pocket PC 2002 ARM

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

- Microsoft Windows 2000

  Compliance is maintained on platforms for which the binary executable remains unchanged including (but not limited to):
  - Microsoft Windows NT 4
  - Microsoft Windows XP
  - Microsoft Windows 2003.

Refer to the NIST document, *Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program*, for resolution on the issue of "Multi user" modes. This document is located at:
http://csrc.nist.gov/cryptval/140-1/FIPS1402IG.pdf.

Module Interfaces

The Cryptographic Module is tested as a multi-chip, standalone, module. The Cryptographic Module's physical interfaces consist of the keyboard, mouse, monitor, CD-ROM drive, floppy drive, serial ports, USB ports, COM ports, and network adapter(s). However, the module sends/receives data entirely through the underlying logical interface, a C-language API documented in the Cryptographic Module API Reference. The module provides for Control Input through the API calls. Data Input and Output are provided in the variables passed with API calls, and Status Output is provided through the returns, exceptions, and error codes that are documented for each call.

Roles and Services

The Cryptographic Module meets all FIPS 140-2 level 1 requirements for Roles and Services, implementing both a User role and Crypto Officer role. As allowed by FIPS 140-2, the Cryptographic Module does not support user identification or authentication for these roles. Only one role may be active at a time and the Cryptographic Module does not allow concurrent operators.

At the highest level, the services provided by the module include:

- signing into and out of the Crypto Officer role
- running the self-tests
- querying for status on the last self-tests executed
- symmetric key processing
- asymmetric key processing
- hashing and digital signature support.

A complete description of all of the module's services is provided in Services on page 13.

Crypto Officer Role

An operator assuming the Crypto Officer role can call any of the module's functions. The complete list of the functionality available to the Crypto Officer is outlined in Services on page 13.

User Role

An operator assuming the User role can utilize the entire Cryptographic Module API except for the Startup Self Test Function, which is reserved for the Crypto Officer. The API and its functions and capabilities are documented in Services on page 13.

Cryptographic Key Management

Key Generation

The Cryptographic Module supports generation of DSA, RSA, and Diffie-Hellman (DH) public and private keys. Furthermore, the module employs a FIPS 186-2 random number generator using SHA-1 for generating symmetric keys used in algorithms such as AES, DES, or TDES.

Key Storage

Public and private keys are provided to the Cryptographic Module by the operator, and their state is destroyed when the operator indicates the key is finished being used. No persistent storage of keys is handled. The Cryptographic Module does not provide long-term cryptographic key storage. If an operator chooses to store keys, the operator is responsible for storing keys exported from the module.

Key Access

An authorized operator of the module has access to all key data created during the module's operation.

Key Protection/Zeroization

All key data resides in internally allocated data structures and can be output only using the module's defined API. MSWin protects memory and process space from unauthorized access. The operator should follow the steps outlined in the Cryptographic Module *API Reference* and *Developer's Guide* to ensure sensitive data is protected by freeing the data from memory when it is no longer needed.

Cryptographic Algorithms

The Cryptographic Module supports a wide variety of cryptographic algorithms (see the following table for a list of algorithms supported by the Cryptographic Module). FIPS 140-2 requires that FIPS-approved algorithms be used whenever there is an applicable FIPS standard. Thus, as the following table summarizes, only a subset of the algorithms provided by the Cryptographic Module may be used in compliance with the FIPS 140-2 requirements. For more information on using the Cryptographic Module in a FIPS compliant manner, refer to Secure Operation of the Cryptographic Module on page 11.

Type	Algorithm	FIPS-Approved
Public Key	Diffie-Hellman (DH)	No[a]
	DSA (key sizes: 512-4096)	Yes (FIPS 186-2)
	RSA (key sizes: 512-8192)	Yes (FIPS 186-2)
Symmetric Key	AES (CBC, CFB, ECB, OFB)	Yes (FIPS 197)
	DES (CBC, CFB, ECB, OFB)	Yes (FIPS 46-3)
	RC2 (CBC, CFB, ECB, OFB)	No
	RC4 (CBC, CFB, ECB, OFB)	No
	RC5 (CBC, CFB, ECB, OFB)	No
	TDES (CBC, CFB, ECB, OFB)	Yes (FIPS 46-3)
Digest	MD2	No
	MD5	No
	SHA-1	Yes (FIPS 180-1)
	SHA-2	No
MAC (Message Authentication Code)	SHA-1 HMAC	Yes (FIPS 198a)
	MD5 HMAC	No
PRNG (Pseudo Random Number Generator)	FIPS 186-2	Yes (FIPS 186-2)

a. *DH is not a FIPS approved algorithm but is allowed for use in FIPS mode, as it is a commercially available public-key based key distribution technique.*

Self-Test

The Cryptographic Module performs power-up and conditional self-tests to ensure proper operation.

Power-Up Self-Tests

The power-up self-tests implemented in the Cryptographic Module include known answer tests (KAT) for AES, DES, TDES, SHA-1, DSA, and RSA. Also executed at power-up is a PRNG KAT and a software/firmware integrity check. Power-up self-tests are executed automatically when the module is loaded.

Conditional Self-Tests

The Cryptographic Module performs two conditional self-tests: a pair-wise consistency test each time the module generates a DSA, DH, or RSA public/private key pair, and a continuous random number generator test each time the module produces random data per its FIPS 186-2 random number generator.

Mitigation of Other Attacks

RSA key operations implement blinding by default, providing a defence against timing attacks. Blinding is implemented through blinding modes, and the following options are available:

- Blinding mode off
- Blinding mode with no update, where the blinding value is constant for each operation
- Blinding mode with full update, where a new blinding value is used for each operation.

Secure Operation of the Cryptographic Module

This section provides an overview of cryptography, including the concepts of authentication, integrity and privacy. The following topics are discussed:

- Operating the Cryptographic Module
- Modes of Operation.

Operating the Cryptographic Module

The Cryptographic Module may be placed into FIPS mode by calling the `CRYPTOC_FIPS140_enable_nist_operating_mode()` function. After making the `CRYPTOC_FIPS140_enable_nist_operating_mode()` function call, the Cryptographic Module enforces that only the FIPS approved algorithms listed in the section, Cryptographic Algorithms on page 9, are available to operators. To disable FIPS mode, call `CRYPTOC_FIPS140_enable_non_nist_operating_mode()`.

Modes of Operation

There are four modes of FIPS 140 operation: `DISABLED_MODE`, `FIPS140_MODE`, `NON_FIPS140_MODE`, and `FIPS140_SSL_MODE`. Use the functions listed after each mode below to enter and to check that the module is in the specified mode.

DISABLED MODE

This mode indicates that the FIPS140 library is disabled, usually due to an internal or caller's usage error. No future transition into `FIPS140_MODE` or `NON_FIPS140_MODE` is permitted. The caller's current operating system process may continue to operate with the currently opened library and cryptographic contexts, but no additional contexts may be opened.

- `CRYPTOC_FIPS140_disable_operating_modes()`
- `CRYPTOC_FIPS140_operating_mode_is_disabled()`.

FIPS 140 MODE

This mode indicates that the FIPS140 library is running in FIPS140_MODE. A transition into NON_FIPS140_MODE is possible only after all FIPS140_MODE library contexts have been closed.

- CRYPTOC_FIPS140_enable_fips140_operating_mode()
- CRYPTOC_FIPS140_operating_mode_is_fips140().

NON FIPS 140 MODE

This mode indicates that the FIPS140 library is running in NON_FIPS140_MODE. A transition into FIPS140_MODE is possible only after all NON_FIPS140_MODE library contexts have been closed.

- CRYPTOC_FIPS140_enable_non_fips140_operating_mode()
- CRYPTOC_FIPS140_operating_mode_is_non_fips140().

FIPS 140 SSL MODE

This mode indicates that the FIPS140 library is running in FIPS140_SSL_MODE. A transition into NON_FIPS140_MODE is possible only after all FIPS140_SSL_MODE library contexts have been closed.

FIPS140_SSL_MODE is FIPS140_MODE with the addition of those items required to perform TLS in a FIPS140-compatible manner.

- CRYPTOC_FIPS140_enable_fips140_ssl_operating_mode()
- CRYPTOC_FIPS140_operating_mode_is_fips140_ssl().

Services

Operating modes control which cryptographic services are available to the application. For more information see the *FIPS Specific Module Operations* document and the Cryptographic Module-specific *API Reference*. The following topics are included in this section:

- Initialization
- Operating Controls
- Roles
- Self-Tests
- Library Controls
- DSA Parameter Controls
- Crypto Commands
- Low-Level Random Routines
- Public Key Operations
- Symmetric Key Operations.

Initialization

Calls to initialize the library and pull in desired cryptographic features:

- `CRYPTOC_ME_FIPS140_library_init()`
- `CRYPTOC_ME_FIPS140_library_free()`.

Operating Controls

Operator modes control which cryptographic services are available to the application:

- `CRYPTOC_FIPS140_operating_mode_is_disabled()`
- `CRYPTOC_FIPS140_enable_non_fips140_operating_mode()`
- `CRYYTOC_FIPS140_operating_mode_is_non_fips140()`
- `CRYPTOC_FIPS140_enable_fips140_operating_mode()`
- `CRYPTOC_FIPS140_operating_mode_is_fips140()`
- `CRYPTOC_FIPS140_enable_operating_mode_fips140_ssl()`
- `CRYPTOC_FIPS140_operating_mode_is_fips140_ssl()`.

Roles

These procedures control the assumed roles:

- `CRYPTOC_FIPS140_sign_in_state_is_disabled()`
- `CRYPTOC_FIPS140_set_user_sign_in_state()`
- `CRYPTOC_FIPS140_sign_in_state_is_user()`
- `CRYPTOC_FIPS140_set_officer_sign_in_state()`
- `CRYPTOC_FIPS140_sign_in_state_is_officer()`.

Self-Tests

These are the self-test programs:

- `CRYPTOC_FIPS140_me_fips140_startup_self_test()` (Note: This function is available to the Crypto Officer only)
- `CRYPTOC_FIPS140_get_self_test_result()`.

Library Controls

The `CRYPTOC_FIPS140_get_version` function returns the version string for the Cryptographic Module. This version is independent from the product library version returned by the toolkit which can be queried via the `CRYPTOC_ME_library_info` function.

DSA Parameter Controls

Export NIST-DSA Parameter-Generation API:

- `DSA_generate_parameters()`
- `DSA_enable_default_method()`
- `DSA_free()`
- `DSA_is_prime()`.

Crypto Commands

The following exports are part of the Cryptographic Module API:

- R_CR_asym_decrypt()
- R_CR_asym_decrypt_init()
- R_CR_asym_encrypt()
- R_CR_asym_encrypt_init()
- R_CR_CTX_get_info()
- R_CR_CTX_set_info()
- R_CR_decrypt()
- R_CR_decrypt_final()
- R_CR_decrypt_init()
- R_CR_decrypt_update()
- R_CR_digest()
- R_CR_digest_final()
- R_CR_digest_init()
- R_CR_digest_update()
- R_CR_dup()
- R_CR_encrypt()
- R_CR_encrypt_final()
- R_CR_encrypt_init()
- R_CR_encrypt_update()
- R_CR_free()
- R_CR_generate_key()
- R_CR_generate_key_init()
- R_CR_generate_parameter()
- R_CR_generate_parameter_init()
- R_CR_get_default_method()
- R_CR_get_error_string()
- R_CR_get_info()
- R_CR_key_exchange_init()
- R_CR_key_exchange_phase_1()
- R_CR_key_exchange_phase_2()

- R_CR_mac()
- R_CR_mac_final()
- R_CR_mac_init()
- R_CR_mac_update()
- R_CR_new()
- R_CR_random_bytes()
- R_CR_random_seed()
- R_CR_set_info()
- R_CR_sign()
- R_CR_sign_final()
- R_CR_sign_init()
- R_CR_sign_update()
- R_CR_verify()
- R_CR_verify_final()
- R_CR_verify_init()
- R_CR_verify_mac()
- R_CR_verify_mac_final()
- R_CR_verify_mac_init()
- R_CR_verify_mac_update()
- R_CR_verify_update().

Low-Level Random Routines

- R_rand_meth_sha1()
- R_rand_add_entropy()
- R_rand_bytes()
- R_rand_entropy_count()
- R_rand_get_default()
- R_rand_entropy_func()
- R_rand_lib_cleanup()
- R_rand_load_file()
- R_rand_seed()
- R_rand_set_default()
- R_rand_entropy_func()
- R_rand_write_file().

Public Key Operations

- R_PKEY_new()
- R_PKEY_free()
- R_PKEY_from_binary()
- R_PKEY_get_info()
- R_PKEY_set_info()
- R_PKEY_CTX_free()
- R_PKEY_CTX_new().

Symmetric Key Operations

- R_SKEY_free()
- R_SKEY_get_info()
- R_SKEY_set_info()
- R_SKEY_new().

Acronym List

Following is a list of acronyms used in this document, each followed by a definition:

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
CBC	Cipher-block Chaining
CFB	Cipher Feedback
DES	Data Encryption Standard
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
FIPS	Federal Information Processing Standard
KAT	Known Answer Test
MD2	Message Digest Algorithm 2
MD5	Message Digest Algorithm 5
NIST	National Institute of Standards and Technology
OFB	Output Feedback
RC2	Rivest's Code 2
RC4	Rivest's Code 4
RC5	Rivest's Code 5
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm
TDES	Triple-DES
TLS	Transport Layer Security