RSA BSAFE®

Crypto-C ME

Security Policy

Version 2.0

December 8, 2005

Powerful cryptography for the smallest of devices

Contact Information

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

RSA Security Inc.	RSA Security Ireland Limited
www.rsasecurity.com	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.

This product includes patented technology licensed from Entrust Technologies Inc. (US Patent# 5,699,431).

Published December 8, 2005

001-001004-200-002-000

Table of Contents

1. 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/.

2. References

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

- The RSA website contains information on their full line of products and services at http://www.rsasecurity.com/.
- An overview of the Crypto-C ME module is located at http://www.rsasecurity.com/node.asp?id=1210.
- The RSA BSAFE product overview is provided at http://www.rsasecurity.com/node.asp?id=1202.
- For answers to technical or sales related questions please refer to http://www.rsasecurity.com/contact/.

3. 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. Crypto-C ME Module on 5 describes the Cryptographic Module and how it meets FIPS 140-2 requirements. Secure Operation of the Cryptographic Module on page 12 specifically addresses the required configuration for the FIPS-mode of operation. Services on page 15 lists all of the functions provided by the Cryptographic Module. Acronyms/Definitions on page 18 lists the definitions for the acronyms used in this document.

4. Crypto-C ME Module

This section provides an overview of the Crypto-C ME Module. The following topics are discussed:

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

4.1. 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 (Note: When operating in a FIPS-approved manner, the set of algorithm implementations is not customizable).

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.

4.2. 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 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 7.2 x86 (32-bit)
  Compliance is maintained on platforms for which the binary executable remains unchanged including (but not limited to):
  - Red Hat Linux 7.1, and 8.0
  - Red Hat Advanced Server 2.1.

- Red Hat Enterprise Linux AS 3.0 x86 (32-bit)
  Compliance is maintained on platforms for which the binary executable remains unchanged.

- 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 (Sun OS 5.9) Sparc V8 (32-bit).
  - Sun Microsystems Solaris 10 (Sun OS 5.10) Sparc V8 (32-bit).

- 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 (Sun OS 5.9) Sparc V8+ (32-bit).
  - Sun Microsystems Solaris 10 (Sun OS 5.10) Sparc V8+ (32-bit).

- 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 (Sun OS 5.9) Sparc V9 (64-bit).
  - Sun Microsystems Solaris 10 (Sun OS 5.10) Sparc V9 (64-bit).

- Microsoft Windows Mobile 2003 (WinCE 4.20) for Pocket PC 32-bit ARM

- Microsoft Windows XP Service Pack 2
  Compliance is maintained on platforms for which the binary executable remains unchanged including (but not limited to):
  - Microsoft Windows NT 4
  - Microsoft Windows 2000 Service Pack 4
  - Microsoft Windows 2003 Server.

- IBM AIX 5L v5.3 (32bit)
  Compliance is maintained on platforms for which the binary executable remains unchanged including (but not limited to):
  - IBM AIX 5L v5.2 (32bit)

- HP-UX 11.23 Itanium 2 (64-bit)
  Compliance is maintained on platforms for which the binary executable remains unchanged.

- HP-UX 11.11 PA-RISC 2.0 (32-bit)
  Compliance is maintained on platforms for which the binary executable remains unchanged including (but not limited to):
  - HP-UX 11.11 through 11.23 for PA-RISC 2.0 processors.

- HP-UX 11.23 PA-RISC 2.0W (64-bit)
  Compliance is maintained on platforms for which the binary executable remains unchanged including (but not limited to):
  - HP-UX 11.11 through 11.23 for PA-RISC 2.0W processors.
- VxWorks 5.4 PPC 604 (32-bit)
- VxWorks 5.5 PPC 603 (32-bit)
- VxWorks 5.5 PPC 604 (32-bit)
  -

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.

4.3. Module Interfaces

The Cryptographic Module is evaluated 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.

4.4. Roles and Services

The Crypto-C ME module meets all FIPS140-2 level 1 requirements for Roles and Services, implementing both a User (User) role and Officer (CO) role. As allowed by FIPS 140-2, the Crypto-C ME module does not support user identification or authentication for these roles. Only one role may be active at a time and the Crypto-C ME module does not allow concurrent operators.

Table 1. Crypto-C ME roles and services.

Role	Services
Officer	The Officer has access to a superset of the services that are available to the User. The Officer role may also invoke the full set of self tests inside the module.
User	The User may perform general security functions as described in the Crypto-C ME Developer's Guide. The User may also call specific FIPS 140 module functions as defined in Crypto-C ME Developer's Guide.

4.4.1. Officer Role

An operator assuming the Officer role can call any of the module's functions. The complete list of the functionality available to the Officer is outlined in the Services section on page 15.

4.4.2. User Role

An operator assuming the User role can utilize the entire Crypto-C ME API except for the `R_FIPS140_self_test_full()` method, which is reserved for the Officer. The Crypto-C ME API functions are documented in the Services section on page 15.

4.5. Cryptographic Key Management

4.5.1. Key Generation

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

4.5.2. Key Storage

The Cryptographic Module does not provide long-term cryptographic key storage. If a User chooses to store keys, the User is responsible for storing keys returned to the calling application.

Volatile (short term) memory storage of cryptographic keys employed by the cryptographic module is handled in the following manner:

- The User & Officer roles have equal and complete access to all keys listed in Table 2.

Table 2. Crypto-C ME Key Storage.

Item	Storage
AES keys	In volatile memory only (plaintext)
Triple DES keys	In volatile memory only (plaintext)
HMAC with SHA1 and SHA2 keys	In volatile memory only (plaintext)
Diffie-Hellman public key	In volatile memory only (plaintext)
Diffie-Hellman private key	In volatile memory only (plaintext)
ECC public key	In volatile memory only (plaintext)
ECC private key	In volatile memory only (plaintext)
RSA public key	In volatile memory only (plaintext)
RSA private key	In volatile memory only (plaintext)
DSA public key	In volatile memory only (plaintext)
DSA private key	In volatile memory only (plaintext)
PRNG seeds(FIPS 186-2)	In volatile memory only (plaintext)

4.5.3. Key Access

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

4.5.4. Key Protection/Zeroization

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

4.6. Cryptographic Algorithms

The Crypto-C ME module supports a wide variety of cryptographic algorithms. FIPS 140-2 requires that FIPS-approved algorithms be used whenever there is an applicable FIPS standard. The following table lists the FIPS approved algorithms supported by the Crypto-C ME module.

Table 3. Crypto-C ME algorithms allowed in FIPS mode

Algorithm	Validation Certificate
AES ECB, CBC, CFB (128), OFB (128), CTR – [128, 192, 256 bit key sizes]	Cert. 303
AES CCM	Cert. 7
3DES ECB, CBC, CFB (64bit) , and OFB (64 bit)	Cert. 378
Diffie-Hellman	Non-Approved (Allowed in FIPS mode)
DSA	Cert. 143
EC-Diffie-Hellman	Non-Approved (Allowed in FIPS mode)
EC-DSA, EC-DSA-SHA1	Cert. 11
FIPS 186-2 PRNG (Change Notice 1-with and without the mod q step)	Cert. 130
RSA X9.31, PKCS#1 V.1.5, PKCS#1 V.2.1 (SHA256 - PSS)	Cert. 96
RSA encrypt/decrypt	Non-Approved (Allowed in FIPS mode for key transport)
SHA-1	Cert. 380
SHA-224, 256, 384, 512	Cert. 380
HMAC-SHA1, SHA224, SHA256, SHA384, SHA512	Cert. 113

Table 4 - Crypto-C ME Non-FIPS approved algorithms

Algorithm
MD2
MD5
HMAC MD5
DES (non-compliant)
DES40
RC2
RC4
RC5
ECAES
ECDRBG
RSA PKCS#1 V.2.0 (SHA256 - OAEP)

For more information on using Crypto-C ME in a FIPS compliant manner refer to Secure Operation of the Cryptographic Module on page 12.

4.7. Self-Test

The Crypto-C ME module performs a number of power-up and conditional self-tests to ensure proper operation.

4.7.1. Power-Up Self-Tests

The power-up self-tests implemented in the Crypto-C ME module are:

- AES Known Answer Tests (KATs)
- AES CCM Known Answer Tests (KATs)
- TDES KATs
- DES KATs
- SHA-1 KATs
- SHA-224 KATs
- SHA-256 KATs
- SHA-384 KATs
- SHA-512 KATs
- HMAC SHA-1 KATs
- HMAC SHA-224 KATs
- HMAC SHA-256 KATs
- HMAC SHA-384 KATs
- HMAC SHA-512 KATs
- RSA Sign/verify Tests
- DSA Sign/verify Test
- DH conditional test
- ECDSA Sign/verify Test
- PRNG KATs
- Software integrity test

Power-up self-tests are executed automatically when the module is loaded into memory.

4.7.2. Conditional Self-Tests

The Crypto-C ME module performs two conditional self-tests: a pair-wise consistency test each time the module generates a DSA, DH, RSA, or EC 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, ECDRBG and the OTP RNG..

4.7.3. Critical Functions Test

When operating in FIPS140_SSL_MODE, a known answer test is performed for MD5 and HMAC-MD5.

When operating in FIPS140_ECC_MODE, a known answer test is performed for ECAES and ECDRBG.

When operating in FIPS140_SSL ECC_MODE, a known answer test is performed for MD5, HMAC-MD5, ECAES and ECDRBG.

4.7.4. Mitigation of Other Attacks

RSA key operations implement blinding by default, providing a defense 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.

5. Secure Operation of the Cryptographic Module

This section provides an overview of how to securely operate the Crypto-C ME module in order to be in compliance with the FIPS140-2 standards.

5.1. Approved Key Sizes

To use the module in the Approved mode, the following key size ranges must be used:

- DSA key-pair modulus sizes should be between 512 and 1024 bits
- RSA[1] modulus size must be at least 1024 bits
- Diffie-Hellman[2] (DH) modulus size must be at least 1024 bits
- ECDH[3] f values supported are between 163 and 571

5.2. Operating the Cryptographic Module

The Cryptographic Module may be placed into FIPS mode by calling the `R_FIPS140_set_mode()` function with the mode identifier R_FIPS140_MODE_FIPS140. After making the `R_FIPS140_set_mode()` function call, the Cryptographic Module enforces that only the FIPS approved algorithms listed in the Services section on page 15 are available to operators. To disable FIPS mode, call `R_FIPS140_set_mode`() with the mode identifier R_FIPS140_MODE_NON_FIPS140.

The following Services are restricted to operation by the Officer:

- `R_FIPS140_self_tests_full()`

The user of the Cryptographic Module shall link with the static library for their platform which will load the cryptographic module's shared library or dynamic link library at runtime. For additional details see the "FIPS 140-2 Library and Modes of Operation" section in the Crypto-C ME Developers Guide.

5.3. Modes of Operation

There are six modes of operation: `R_FIPS140_MODE_DISABLED`, `R_FIPS140_MODE_FIPS140`, `R_FIPS140_MODE_FIPS140_ECC`, `R_FIPS140_MODE_FIPS140_SSL_ECC`, `R_FIPS140_MODE_FIPS140_SSL`, and `R_FIPS140_NON_FIPS140`. Use the functions listed after each mode below to enter and to check that the module is in the specified mode.

[1] When used for transporting keys and using the minimum allowed modulus size; the minimum strength of encryption provided is 80 bits.

[2] Using the minimum allowed modulus size; the minimum strength of encryption provided is 80 bits.

[3] Provides between 80 and 285 bits of security

Cryptographic keys must not be shared between R_FIPS140_MODE_FIPS140/R_FIPS140_MODE_FIPS140_SSL/R_FIPS140_MODE_FIPS140_ECC /R_FIPS140_MODE_FIPS140_SSL_ECC and R_FIPS140_MODE_DISABLED/R_FIPS140_MODE_NON_FIPS140.

5.3.1. 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 R_FIPS140_MODE_FIPS140, R_FIPS140_MODE_FIPS140_SSL, R_FIPS140_MODE_FIPS140_ECC, R_FIPS140_MODE_FIPS140_SSL_ECC or R_FIPS140_MODE_NON_FIPS140 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.

- R_FIPS140_set_mode() with the mode identifier R_FIPS140_MODE_DISABLED
- R_FIPS140_get_mode() to determine the mode of operation.

5.3.2. FIPS 140 MODE

This mode indicates that the FIPS140 library is running in R_FIPS140_MODE_FIPS140. A transition into R_FIPS140_MODE_NON_FIPS140 shall only be made after all R_FIPS140_MODE_FIPS140 library contexts have been closed.

- R_FIPS140_set_mode() with the mode identifier R_FIPS140_MODE_FIPS140
- R_FIPS140_get_mode() to determine the mode of operation.

5.3.3. FIPS 140 ECC Mode

This mode indicates that the FIPS140 library is running in R_FIPS140_MODE_FIPS140_ECC. A transition into R_FIPS140_MODE_NON_FIPS140 shall only be made after all R_FIPS140_MODE_FIPS140_ECC library contexts have been closed.

- R_FIPS140_set_mode() with the mode identifier R_FIPS140_MODE_FIPS140_ECC
- R_FIPS140_get_mode() to determine the mode of operation.

5.3.4. FIPS 140 SSL ECC Mode

This mode indicates that the FIPS140 library is running in R_FIPS140_MODE_FIPS140_SSL_ECC. A transition into R_FIPS140_MODE_NON_FIPS140 shall only be made after all R_FIPS140_MODE_FIPS140_SSL_ECC library contexts have been closed.

- R_FIPS140_set_mode() with the mode identifier R_FIPS140_MODE_FIPS140_SSL_ECC
- R_FIPS140_get_mode() to determine the mode of operation.

5.3.5. FIPS 140 SSL MODE

This mode indicates that the FIPS140 library is running in R_FIPS140_MODE_FIPS140_SSL. A transition into R_FIPS140_MODE_NON_FIPS140 shall only be made after all R_FIPS140_MODE_FIPS140_SSL library contexts have been closed.

R_FIPS140_MODE_FIPS140_SSL is R_FIPS140_MODE_FIPS140 with the addition of those items required to perform TLS in a FIPS140-compatible manner.

- R_FIPS140_set_mode() with the mode identifier R_FIPS140_MODE_FIPS140_SSL

- `R_FIPS140_get_mode`() to determine the mode of operation.

5.3.6. NON FIPS 140 MODE

This mode indicates that the FIPS140 library is running in `R_FIPS140_MODE_NON_FIPS140`. A transition into `R_FIPS140_MODE_FIPS140` shall only be made after all `R_FIPS140_MODE_NON_FIPS140` library contexts have been closed.

- `R_FIPS140_set_mode`() with the mode identifier `R_FIPS140_MODE_NON_FIPS140`
- `R_FIPS140_get_mode`() to determine the mode of operation.

6. Services

The Cryptographic Module provides the following services. For details of the operation of each of these services see the Developers Guide.

Table 2 - Crypto-C ME Services

Function	Function
BIO_append_filename	R_CR_TYPE_to_string
BIO_clear_flags	R_CR_verify
BIO_clear_retry_flags	R_CR_verify_final
BIO_copy_next_retry	R_CR_verify_init
BIO_debug_cb	R_CR_verify_mac
BIO_dump	R_CR_verify_mac_final
BIO_dump_format	R_CR_verify_mac_init
BIO_dup_chain	R_CR_verify_mac_update
BIO_end_of_msg	R_CR_verify_update
BIO_f_buffer	R_FIPS140_free
BIO_f_null	R_FIPS140_get_default
BIO_find_type	R_FIPS140_get_failure_reason
BIO_flush	R_FIPS140_get_failure_reason_string
BIO_free	R_FIPS140_get_info
BIO_free_all	R_FIPS140_get_interface_version
BIO_get_cb	R_FIPS140_get_mode
BIO_get_cb_arg	R_FIPS140_get_role
BIO_get_close	R_FIPS140_get_supported_interfaces
BIO_get_flags	R_FIPS140_library_free
BIO_get_fp	R_FIPS140_library_init
BIO_get_retry_BIO	R_FIPS140_load_module
BIO_get_retry_reason	R_FIPS140_new
BIO_gets	R_FIPS140_self_tests_full
BIO_method_name	R_FIPS140_self_tests_short
BIO_method_type	R_FIPS140_set_info
BIO_new	R_FIPS140_set_interface_version
BIO_new_file	R_FIPS140_set_mode
BIO_new_fp	R_FIPS140_set_role
BIO_new_mem	R_FIPS140_unload_module
BIO_open_file	R_FORMAT_from_string
BIO_pop	R_FORMAT_to_string
BIO_print_hex	R_free
BIO_printf	R_get_mem_functions
BIO_push	R_LIB_CTX_free
BIO_puts	R_LIB_CTX_get_info
BIO_read	R_LIB_CTX_new
BIO_read_filename	R_LIB_CTX_set_info
BIO_reference_inc	R_lock_ctrl
BIO_reset	R_lock_get_cb
BIO_rw_filename	R_lock_get_name
BIO_s_file	R_lock_num
BIO_s_mem	R_lock_r
BIO_s_null	R_lock_set_cb
BIO_seek	R_lock_w
BIO_set_alg	R_locked_add
BIO_set_asym_key	R_locked_add_get_cb

Function	Function
BIO_set_bio_cb	R_locked_add_set_cb
BIO_set_cb	R_lockid_new
BIO_set_cb_arg	R_lockids_free
BIO_set_cert	R_malloc
BIO_set_close	R_PKEY_cmp
BIO_set_content_type	R_PKEY_CTX_free
BIO_set_flags	R_PKEY_CTX_get_info
BIO_set_fp	R_PKEY_CTX_get_LIB_CTX
BIO_set_store	R_PKEY_CTX_new
BIO_set_unwrapped	R_PKEY_CTX_set_info
BIO_set_verification	R_PKEY_FORMAT_from_string
BIO_should_io_special	R_PKEY_FORMAT_to_string
BIO_should_read	R_PKEY_free
BIO_should_retry	R_PKEY_from_binary
BIO_should_write	R_PKEY_from_public_key_binary
BIO_tell	R_PKEY_get_info
BIO_write	R_PKEY_get_num_bits
BIO_write_filename	R_PKEY_get_num_primes
R_BASE64_decode	R_PKEY_get_PKEY_CTX
R_BASE64_encode	R_PKEY_get_type
R_CR_asym_decrypt	R_PKEY_iterate_fields
R_CR_asym_decrypt_init	R_PKEY_METHOD_free
R_CR_asym_encrypt	R_PKEY_METHOD_get_flag
R_CR_asym_encrypt_init	R_PKEY_METHOD_get_name
R_CR_CTX_alg_supported	R_PKEY_METHOD_get_type
R_CR_CTX_free	R_PKEY_new
R_CR_CTX_get_info	R_PKEY_pk_method
R_CR_CTX_ids_from_sig_id	R_PKEY_print
R_CR_CTX_ids_to_sig_id	R_PKEY_public_cmp
R_CR_CTX_new	R_PKEY_reference_inc
R_CR_CTX_set_info	R_PKEY_set_info
R_CR_decrypt	R_PKEY_to_binary
R_CR_decrypt_final	R_PKEY_to_bio
R_CR_decrypt_init	R_PKEY_to_public_key_binary
R_CR_decrypt_update	R_PKEY_TYPE_from_string
R_CR_DEFINE_CUSTOM_CIPHER_LIST	R_PKEY_TYPE_to_string
R_CR_DEFINE_CUSTOM_METHOD_TABLE	R_rand_add_entropy
R_CR_digest	R_rand_bytes
R_CR_digest_final	R_rand_entropy_count
R_CR_digest_init	R_rand_file_name
R_CR_digest_update	R_rand_free
R_CR_dup	R_rand_get_default
R_CR_encrypt	R_rand_get_entropy_func
R_CR_encrypt_final	R_rand_lib_cleanup
R_CR_encrypt_init	R_rand_load_file
R_CR_encrypt_update	R_rand_new
R_CR_free	R_rand_seed
R_CR_generate_key	R_rand_set_default
R_CR_generate_key_init	R_rand_set_entropy_func
R_CR_generate_parameter	R_rand_write_file
R_CR_generate_parameter_init	R_realloc
R_CR_get_default_imp_method	R_remalloc
R_CR_get_default_method	R_RES_LIST_get_item
R_CR_get_default_signature_map	R_RES_LIST_get_resource
R_CR_get_detail	R_RES_LIST_set_item

Function	Function
R_CR_get_detail_string	R_RES_LIST_set_resource
R_CR_get_detail_string_table	R_set_mem_functions
R_CR_get_error	R_SKEY_free
R_CR_get_error_string	R_SKEY_get_info
R_CR_get_file	R_SKEY_new
R_CR_get_function	R_SKEY_set_info
R_CR_get_function_string	R_TIME_cmp
R_CR_get_function_string_table	R_TIME_CTX_free
R_CR_get_info	R_TIME_CTX_new
R_CR_get_line	R_TIME_dup
R_CR_get_reason	R_TIME_export
R_CR_get_reason_string	R_TIME_free
R_CR_get_reason_string_table	R_TIME_get_time_mi_method
R_CR_ID_from_string	R_TIME_get_utc_time_method
R_CR_ID_to_string	R_TIME_import
R_CR_key_exchange_init	R_TIME_new
R_CR_key_exchange_phase_1	R_TIME_offset
R_CR_key_exchange_phase_2	R_TIME_time
R_CR_mac	R_unlock_r
R_CR_mac_final	R_unlock_w
R_CR_mac_init	RAND_bytes
R_CR_mac_update	RAND_cleanup
R_CR_new	RAND_default_method
R_CR_random_bytes	RAND_file_name
R_CR_random_seed	RAND_get_rand_method
R_CR_RES_CRYPTO_CUSTOM_METHOD	RAND_load_file
R_CR_set_info	RAND_seed
R_CR_sign	RAND_set_rand_method
R_CR_sign_final	RAND_write_file
R_CR_sign_init	
R_CR_sign_update	
R_CR_SUB_from_string	
R_CR_SUB_to_string	
R_CR_TYPE_from_string	

7. Acronyms/Definitions

The following table gives an explanation of the terms and acronyms used throughout this document.

Term	Description
AES	Advanced Encryption Standard. A fast block cipher with a 128-bit block, and keys of lengths 128, 192 and 256 bits. This will replace 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.
DES	Data Encryption Standard. A symmetric encryption algorithm with a 56-bit key. See also Triple DES.
Diffie-Hellman	The Diffie-Hellman asymmetric key exchange algorithm. There are many variants, but typically two entities exchange some public information (for example, public keys or random values) and combines them with their own private keys to generate a shared session key. As private keys are not transmitted, eavesdroppers are not privy to all of the information that composes the session key.
DSA	Digital Signature Algorithm. An asymmetric algorithm for creating digital signatures.
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
Encryption	The transformation of plaintext into an apparently less readable form (called ciphertext) through a mathematical process. The ciphertext may 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
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. Various 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
PC	Personal Computer
PDA	Personal Digital Assistant
PPC	PowerPC
privacy	The state or quality of being secluded from the view and/or presence of others.
private key	The secret key in public key cryptography. Primarily used for decryption but also used for encryption with digital signatures.
PRNG	Pseudo Random Number Generator
RC2	Block cipher developed by Ron Rivest as an alternative to the DES. It has a block size of 64 bits and a variable key size. It is a legacy cipher and RC5 should be used in preference.
RC4	Symmetric algorithm designed by Ron Rivest using variable length keys (usually 40 bit or 128 bit).
RC5	Block cipher designed by Ron Rivest. It is parameterizable in its word size, key length and number of rounds. Typical use involves a block size of 64 bits, a key size of 128 bits and either 16 or 20 iterations of its round function.
RNG	Random Number Generator
RSA	Public key (asymmetric) algorithm providing the ability to encrypt data and create and verify digital signatures. RSA stands for Rivest, Shamir, and Adleman, the developers of the RSA public key cryptosystem.
SHA	Secure Hash Algorithm. An algorithm which creates a unique hash value for each possible input. SHA takes an arbitrary input which is hashed into a 160-bit digest.
SHA-1	A revision to SHA to correct a weakness. It produces 160-bit digests. SHA-1 takes an arbitrary input which is hashed into a 20-byte digest.
SHA-2	The NIST-mandated successor to SHA-1, to complement the Advanced Encryption Standard. It is a family of hash algorithms (SHA-256, SHA-384 and SHA-512) which produce digests of 256, 384 and 512 bits respectively.
Triple DES	A variant of DES which uses three 56-bit keys.

8. Contacting RSA Security

See the RSA Security Web site at http://www.rsasecurity.com for the latest news, security bulletins and information about coming events.

Go to http://www.rsasecurity.com/node.asp?id=1202 for RSA BSAFE product information.

The RSA Laboratories cryptography FAQ at http://www.rsasecurity.com/rsalabs/node.asp?id=2152 contains frequently asked questions.

RSA Developer Central at http://developer.rsasecurity.com enables you to interact with other developers and RSA Security staff, read security-related articles and get answers to security and product questions.

8.1. Support and Service

See http://www.rsasecurity.com./node.asp?id=1067 or https://knowledge.rsasecurity.com if you have any questions or require additional information.

8.2. Purchasing Printed Product Documentation

All documentation for your RSA Security product is included in electronic format on the CD or in the download you have received. You can print product documentation directly from these files if you require a hard copy.

RSA Security also offers customers the option to purchase printed and bound copies of key documents for some products. See http://www.rsasecurity.com/go/documentation for more information.

8.3. Feedback

We welcome your feedback on RSA Security documentation. Please e-mail bsafeuserdocs@rsasecurity.com.