IBM® Crypto for C (ICC)
Version 1.1, 1.2, 1.2.1, and 1.2.2
FIPS 140-2 Non-Proprietary Security Policy, version 1.0
October 1st, 2004

Copyright Notice

Table Of Contents

2. References and Abbreviations

2.1 References

Author	Title
NIST	FIPS PUB 140-2: Security Requirements For Cryptographic Modules, May 2001
NIST	[Derived Test Requirements for FIPS PUB 140-2, November 2001
NIST	Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program
Ed Dobner	IBM Crypto for C (ICC) Version 0.1 Design, Document Version 1 - November 26, 2002

2.2 Abbreviations

AES	The Advanced Encryption Standard. The AES is intended to be issued as a FIPS standard and will replace DES. In January 1997 the AES initiative was announced and in September 1997 the public was invited to propose suitable block ciphers as candidates for the AES. NIST is looking for a cipher that will remain secure well into the next century. NIST selected Rijndael as the AES algorithm.
CMVP	(The NIST) Cryptographic Module Validation Program; an integral part of the Computer Security Division at NIST, the CMVP encompasses validation testing for cryptographic modules and algorithms
Crypto	Cryptographic capability/functionality
CSE	The (Canadian) Communications Security Establishment; An entity operating under the Canadian Department of National Defence, CSE provides technical advice, guidance and services to the Government of Canada to maintain the security of its information and information infrastructures. The CMV Program was established by NIST and CSE in July 1995.
DER	Distinguished Encoding Rules
DES	The Data Encryption Standard, an encryption block cipher defined and endorsed by the U.S. government in 1977 as an official standard; the details can be found in the latest official FIPS (Federal Information Processing Standards) publication concerning DES. It was originally developed at IBM. DES has been extensively studied since its publication and is the most well-known and widely used cryptosystem in the world. As specified in FIPS PUB 46-3, Single DES (i.e., DES) is currently permitted for legacy systems only. New procurements to support legacy systems should, where feasible, use Triple DES products running in the single DES configuration.
DH	Diffie-Hellman key agreement.
DSA	Digital Signature Algorithm - US Federal Information Processing Standard FIPS 186 (Digital Signature Standard, DSS), ANSI X9.30
ICC	IBM Crypto for C-language is a general-purpose cryptographic provider module.
Libcrypt	The cryptography engine of OpenSSL.

MD2 MD4 MD5	MD2, MD4, and MD5 are message-digest algorithms developed by Rivest. They are meant for digital signature applications where a large message has to be "compressed" in a secure manner before being signed with the private key. All three algorithms take a message of arbitrary length and produce a 128-bit message digest. While the structures of these algorithms are somewhat similar, the design of MD2 is quite different from that of MD4 and MD5 and MD2 was optimized for 8-bit machines, whereas MD4 and MD5 were aimed at 32-bit machines. Description and source code for the three algorithms can be found as Internet RFCs 1319 - 1321.
NIST	(The) National Institute of Standards and Technology; NIST is a non-regulatory federal agency within the U.S. Commerce Department's Technology Administration. NIST's mission is to develop and promote measurement, standards, and technology to enhance productivity, facilitate trade, and improve the quality of life. NIST oversees the Cryptographic Module Validation Program.
OpenSSL	A collaborative effort to develop a robust, commercial-grade, full-featured and Open Source toolkit implementing the Secure Socket Layer (SSL V1/V3) and Transport Layer Security (TLS V1) protocols.
RC2	A variable key-size block cipher designed by Rivest for RSA Data Security. "RC" stands for "Ron's Code" or "Rivest's Cipher." It is faster than DES and is designed as a "drop-in" replacement for DES. It can be made more secure or less secure than DES against exhaustive key search by using appropriate key sizes. It has a block size of 64 bits and is about two to three times faster than DES in software. The algorithm is confidential and proprietary to RSA Data Security. RC2 can be used in the same modes as DES.
RC4	A stream cipher designed by Rivest for RSA Data Security. It is a variable key-size stream cipher with byte-oriented operations.
RSA	A public-key cryptosystem for both encryption and authentication; it was invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman.
SHA-1	The Secure Hash Algorithm, the algorithm specified in the Secure Hash Standard (SHS), was developed by NIST and published as a federal information processing standard. SHA-1 was a revision to SHA that was published in 1994. The revision corrected an unpublished flaw in SHA.
Triple DES	Based on the DES standard; the plaintext is, in effect, encrypted three times. Triple DES (TDEA), as specified in ANSI X9.52, is recognized as a FIPS approved algorithm and is the FIPS approved symmetric encryption algorithm of choice.

3. Introduction

This document is a non-proprietary FIPS 140-2 Security Policy for the IBM Crypto for C (ICC), Version 1.1, 1.2, 1.2.1, and 1.2.2 cryptographic module. It contains a specification of the rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-2 (Federal Information Processing Standards Publication 140-2) for a Level 1 multi-chip standalone module. This Policy forms a part of the submission package to the testing lab.

FIPS 140-2 specifies the security requirements for a cryptographic module protecting sensitive information. Based on four security levels for cryptographic modules this standard identifies requirements in eleven sections. For more information about the standard visit http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.

- For more information on the FIPS 140-2 standard and validation program please refer to the NIST website at http://csrc.nist.gov/cryptval/.
- For more information about IBM software please visit http://www.ibm.com

3.1 Purpose of the Security Policy

There are three major reasons that a security policy is required:

- It is required for FIPS 140-2 validation.
- It allows individuals and organizations to determine whether the cryptographic module, as implemented, satisfies the stated security policy.
- It describes the capabilities, protection, and access rights provided by the cryptographic module, allowing individuals and organizations to determine whether it will meet their security requirements.

3.2 Target Audience

This document is intended to be part of the package of documents that are submitted for FIPS validation. It is intended for the following people:

- Developers working on the release
- Product Verification
- Documentation
- Product and Development Managers

4. Cryptographic Module Definition

This section defines the cryptographic module that is being submitted for validation to FIPS PUB 140-2, level 1.

The IBM Crypto for C v1.1, 1.2, 1.2.1, and 1.2.2 (ICC) cryptographic module is implemented in the C programming language. It is packaged as dynamic (shared) libraries usable by applications written in a language that supports C language linking conventions (e.g. C, C++, Java, Assembler, etc.) for use on commercially available operating systems. The ICC allows these applications to access cryptographic functions using an Application Programming Interface (API) provided through an ICC import library and based on the API defined by the OpenSSL group. The cryptographic boundary is defined to be the enclosure of the computer that runs the ICC software.

As outlined in G.5 of the Implementation Guidance for FIPS 140-2, the module maintains its compliance on other operating systems, provided:

- The operating system meets the operational environment requirements at the module's level of validation
- The module does not require modification to run in the new environment

ICC **was** tested and validated on a machine running Microsoft Windows 2000® with Service Pack 3 and Windows 2000® Advanced Server. The software module maintains compliance when running on other versions of Microsoft Windows. ®

ICC **was** tested and validated on a machine running the AIX ® 5.2 operating system. The software module maintains compliance when running on other versions of AIX. ®

ICC **was** tested and validated on a machine running the Solaris® 5.8 operating system. The software module maintains compliance when running on other versions of Solaris. ®

ICC **was** tested and validated on a machine running the SuSE® Linux Linux Enterprise Server 8.0 operating system (x86 and PowerPC). The software module maintains compliance when running on other Linux based operating systems.

ICC **was** tested and validated on a machine running the RedHat Linux Advanced Server 2.1 (x86). The software module maintains compliance when running on other Linux based operating systems.

ICC **was** tested and validated on a machine running the HPUX 11i. The software module maintains compliance when running on other HPUX based operating systems.

ICC **was** tested and validated on a machine running z/Linux 2.4. The software module maintains compliance when running on other z/Linux operating system releases.

5. FIPS 140-2 Specifications

5.1. Ports and Interfaces

The ICC meets the requirements of a multi-chip standalone module. Since the ICC is a software module, its interfaces are defined in terms of the API that it provides. Data Input Interface is defined as those API calls that accept, as their arguments, data to be used or processed by the module. The API calls that return, by means of return value or arguments of appropriate types, data generated or otherwise processed by the module to the caller constitute Data Output Interface. Control Input Interface is comprised of the call used to initiate the module and the API calls used to control the operation of the module.

Status Output Interface is defined as the API calls that provide information about the status of the module. The function ICC_GetStatus, when called, will indicate the status of the ICC module. This may be called anytime after ICC_Init .

5.2 Roles, Services and Authentication

5.2.1 Roles and Authentication

The ICC implements the following two roles: Crypto-Officer role and User role (there is **no** Maintenance Role). The Operating System (OS) provides functionality to require any user to be successfully authenticated prior to using any system services. However, the Module does not support user identification or authentication that would allow for distinguishing users between the two supported roles. Only a single operator assuming a particular role may operate the Module at any particular moment in time. The OS authentication mechanism must be enabled to ensure that none of the Module's services are available to users who do not assume an authorized role.

An operator is explicitly in the User or Crypto Officer role based upon the service(s) chosen. If any of the User specific services are called, then the operator is in the User role; otherwise the operator is in the Crypto Officer role.

The Module does not identify nor authenticate any user (in any role) that is accessing the Module. This is only acceptable for a FIPS 140-2, Security Level 1 validation.

The two roles are defined per the FIPS140-2 standard as follows:

1. **Crypto Officer -** any entity that can access services implemented in the Module and, install and initialize the Module

2. **User** - any entity that can access services implemented in the Module.

Table 1, below, lists the Roles and their associated authentication:

Role	Authentication Type	Authentication Data	Authentication Mechanism	Authentication Strength
Crypto Officer	Not required	Not required	Not required	Not required
User	Not required	Not required	Not required	Not required

Table 1: Roles and associated authentication.

5.2.2 Authorized Services

An operator is explicitly in the User or Cryptographic Officer role based upon the services chosen. If any of the User specific services are called, then the operator is in the User role; otherwise the operator is in the Cryptographic Officer role.

The API's will provide basic encryption and decryption services as follows:

- Hash function
- Symmetric key generation
- Asymmetric key generation
- Random number generation
- Encryption/Decryption with a symmetric key
- Encryption/Decryption with a private asymmetric key
- Encryption/Decryption with a public asymmetric key
- Signing with a private asymmetric key
- Verify signature with a public asymmetric key
- Error reporting
- Random Number Seed
- Diffie-Hellman key agreement

When operating in FIPS approved mode no unapproved algorithms may be used. There is an allowance for key establishment and exchange to use any algorithm when operating in FIPS approved mode (under the phrase "commercially available methods may be used"). The ICC will not limit the algorithms but in the ICC policy it will list the FIPS approved algorithms, the allowances/exceptions (e.g. SSL key exchange and establishment) and the algorithms that are not FIPS approved.

A list of Authorized Services for each role is provided in **table 2**, below.

Role	Authorized Service
Crypto Officer	**Initialization of the Module** • **Key Input/Output** • **CSP Input/Output** - State of Cryptographic module **Symmetric Data Encryption/Decryption** ▪ FIPS Approved: o DES - ECB, CFB, CBC & OFB modes o Triple DES - ECB, CFB, CBC & OFB modes o AES-ECB, CFB, CBC & OFB modes; 128, 192 & 256 Bits ▪ Non FIPS: o RC2-CBC o RC2-ECB o RC2-CFB o RC2-OFB o RC4-40 o RC2-40-CBC o RC2-64-CBC o Blowfish o CAST **Asymmetric Data Encryption/Decryption** ▪ Non-FIPS: o RSA o RSA-NP-MD5 o RSA-SHA1-2 **Digest Algorithms** ▪ FIPS Approved: o SHA1 ▪ Non-FIPS Approved: o MD2 o MD4 o MD5 o RIPEMD **Data Authentication** ▪ FIPS Approved: o RSA ▪ Non-FIPS Approved: o HMAC MD5 o HMAC SHA-1
User	**Symmetric Data Encryption/Decryption** ▪ FIPS Approved: o DES - ECB, CFB, CBC & OFB modes

Role	Authorized Service
	o Triple DES - ECB, CFB, CBC & OFB modes o AES-ECB, CFB, CBC & OFB modes; 128, 192 & 256 Bits ▪ Non FIPS: o RC2-CBC o RC2-ECB o RC2-CFB o RC2-OFB o RC4-40 o RC2-40-CBC o RC2-64-CBC o Blowfish o CAST **Asymmetric Data Encryption/Decryption** ▪ Non-FIPS: o RSA o RSA-NP-MD5 o RSA-SHA1-2 **Digest Algorithms** ▪ FIPS Approved: o SHA1 ▪ Non-FIPS Approved: o MD2 o MD4 o MD5 o RIPEMD **Key Agreement** ▪ FIPS Allowed: o Diffie-Hellman **Digital Signature** ▪ Non-FIPS Approved: o DSA (non-compliant) **Data Authentication** ▪ FIPS Approved: o RSA ▪ Non-FIPS Approved: o HMAC MD5 o HMAC SHA-1

Table 2: Roles and authorized services.

5.2.3 Access Rights Within Services

An operator performing a service within any role can read/write cryptographic keys and critical security parameters (CSP) only through the invocation of a service by use of the Cryptographic Module API. Table 3, below, lists the services corresponding to each role.

Each service within each role can only access the cryptographic keys and CSPs that the service's API defines. The following cases exist:

- A cryptographic key or CSP is provided to an API as an input parameter; this indicates read/write access to that cryptographic key or CSP.
- A cryptographic key or CSP is returned from an API as a return value; this indicates read access to that cryptographic key or CSP.

Service	Cryptographic Keys and CSPs	Types of Access
Symmetric Encryption/Decryption	Symmetric Key	Read/Write
Asymmetric Encryption/Decryption	Asymmetric Key Pair	Read/Write
Digital Signature Generation/Verification	Asymmetric Key Pair	Read/Write
Hash Generation	None	N/A
MAC Generation	Symmetric Key	Read/Write
Key Agreement	Asymmetric Key Pair	Read/Write
Random Number Generation	Seed	N/A
Initialization of the Cryptographic Module	None	N/A
Key Input/Output	Key	Read/Write
CSP Input/Output	CSP	Read/Write

Table 3: Access rights within services.

NOTE: The ICC provides API's to do message digest (Hash). If the application needs an encrypted Hash, it must take the hash value and call the appropriate encryption functions to encrypt the data. The one set of functions that will perform a message digest and encrypt that message digest is the signature creation functions

5.2.4 Operational Rules and Assumptions

The following operational rules must be followed by **any user** of the Module:

1. The Module is to be used by a single human operator at a time and may not be actively shared among operators at any period of time.
2. The OS authentication mechanism must be enabled in order to prevent unauthorized users from being able to access system services.

3. All keys entered into the module must be verified as being legitimate and belonging to the correct entity by software running on the same machine as the module.
4. Since the ICC runs on a general-purpose processor all main data paths of the computer system will contain cryptographic material. The following items need to apply relative to where the ICC will execute:
   - Virtual (paged) memory must be secure (local disk or a secure network).
   - The system bus must be secure.
   - The disk drive that ICC is installed on must be in a secure environment.
5. The above rules must be upheld at all times in order to ensure continued system security and FIPS 140-2 mode compliance after initial setup of the validated configuration. If the module is removed from the above environment, it is assumed to not be operational in the validated mode until such time as it has been returned to the above environment and re-initialized by the user to the validated condition.
6. The DSA algorithm may not be used in the FIPS mode of operation.

NOTE: It is the responsibility of the Crypto-Officer to configure the operating system to operate securely and ensure that only a single operator may operate the Module at any particular moment in time.

The services provided by the Module to a User are effectively delivered through the use of appropriate API calls. In this respect, the same set of services is available to both the User and the Crypto-Officer.

When a client process attempts to load an instance of the Module into memory, the Module runs an integrity test and a number of cryptographic functionality self-tests. If all the tests pass successfully, the Module makes a transition to “FIPS Operation” state, where the API calls can be used by the client to obtain desired cryptographic services. Otherwise, the Module returns to “Error” state and returns an error to the calling application.

5.3 Operational Environment

Along with the conditions stated above in paragraph 5.2.4 (“Operational Rules and Assumptions”), the criteria below must be followed in order to achieve, and maintain, a FIPS 140-2 mode of operation:

5.3.1 Assumptions

The following assumptions are made about the operating environment of the cryptographic module:

1. The prevention of unauthorized reading, writing, or modification of the module's memory space (code and data) by an intruder (human or machine) is assured.

2. The prevention of Replacement or modification of the legitimate cryptographic module code by an intruder (human or machine) is assured.
3. The module is initialized to the FIPS 140-2 mode of operation

5.3.2 Installation and Initialization

The following steps must be performed to install and initialize the module for operating in a FIPS 140-2 compliant manner:

1. The operating system must be configured to operate securely and to prevent remote login. This is accomplished by disabling all services (within the Administrative tools) that provide remote access (e.g. – ftp, telnet, ssh, and server) and disallowing multiple operators to log in at once.
2. The operating system must be configured to allow only a single user. This is accomplished by disabling all user accounts except the administrator. This can be done through the Computer Management window of the operating system. Please see: http://csrc.nist.gov/cryptval/140-1/FIPS1402IG.pdf section 6.1 for the NIST interpretation of “Single User Mode” as applied to servers connected to multiple clients.
3. The module must be initialized to operate in FIPS 140-2 mode; this is done by calling ICC_SetValue with parameter FIPS_APPROVED_MODE and a value of "on"" (Reference ICC design document paragraph 5.3.3).

5.4 Cryptographic Key Management

5.4.1 Implemented Algorithms

The IBM Crypto for C (ICC) version 1.1, 1.2, 1.2.1, and 1.2.2 supports the algorithms (and modes, as applicable) listed above in table 2 in paragraph 5.2.2.

When operating in FIPS approved mode no unapproved algorithms may be used. There is an allowance for key establishment and exchange to use any algorithm when operating in FIPS approved mode (under the phrase “commercially available methods may be used”). The ICC will not limit the algorithms but in the ICC policy it will list the FIPS approved algorithms, the allowances/exceptions (e.g. SSL key exchange and establishment) and the algorithms that are not FIPS approved.

5.4.2 Key Storage

The module does not provide any long-term key storage and no keys are ever stored on the hard disk.

5.4.3 Key Zeroization

Key zeroization is performed via the following API calls:

- ICC_EVP_MD_CTX_cleanup: clears all information from a digest context. It should be called after all operations using a digest are complete so sensitive information does not remain in memory.

- ICC_EVP_CIPHER_CTX_cleanup: clears all information from a cipher context (symmetric keys). It should be called after all operations using a cipher are complete so sensitive information does not remain in memory.
- ICC_RSA_free: frees the structure and its components. The key is erased before the memory is returned to the system. Once this instance has been used with the PKEY functions, it has to be freed with the ICC_EVP_PKEY_free function.
- The random number generator state is cleared automatically when the module is removed from memory.

5.4.4 Random Number Generator

The Random Number Generator (RNG) implemented by the module meet the requirements of FIPS PUB 186-2, Appendix 3.2 (as required in annex C of FIPS PUB 140-2). The RNG Seed has an incremental element/quality (time stamp).

5.5 Self-Tests

The ICC implements a number of self-tests to check proper functioning of the module. This includes power-up self-tests (which are also callable on demand) and conditional self-tests. The self-test can be initiated by calling the function ICC_SelfTest, which returns the operational status of the module (after the self-tests are run) and an error code with description of the error (if applicable).

5.5.1 Show Status

A status function (ICC_GetStatus) when called will indicate the status of the ICC module. This may be called anytime after ICC_Init.

- Get the ICC version
- Inform of FIPS/non FIPS mode
- Error state

5.5.2 Startup Tests

The module performs self-tests automatically when the API function ICC_Attach is called or on demand when the API function ICC_SelfTest is called.

Whenever the startup tests are initiated the module performs the following; if **any** of these tests fail, the module enters the error state.

- **Integrity Test of Digital Signature:** The ICC uses an Integrity test, which is a SHA-1 hash, signed with RSA encryption. At build time a SHA-1 hash is calculated for the ICC dynamic and the libcrypt modules and is included with the static part of the ICC. When the ICC is initialized at run time, the hash values will be given to the ICC dynamic component in order for it to check the two modules.
- **Cryptographic algorithm tests** (known answer tests).
  At startup, a Known Answer Test (encryption & decryption) is performed for the following FIPS approved algorithms:
  - DES - CBC
  - Triple DES - CBC

  - AES - CBC
- **Pairwise Consistency tests are performed for the following:**
  - RSA signature generation test.
  - RSA signature verification test
  - SHA-1 message digest generation test (signed with RSA encryption)

5.5.3 Conditional Tests

- **Pair wise consistency test for public and private key generation.**
  - The consistency of the keys is tested by the calculation and verification of a digital signature. If the digital signature cannot be verified, the test fails.
- **Continuous RNG tests** – The module implements a continuous RNG test as follows:
  - If the call to the RNG produces blocks of n bits (where n > 15), the first n-bit block generated after power-up, initialization, or reset is saved for comparison with the next n-bit block to be generated. Each subsequent generation of an n-bit block is compared with the previously generated block. The test fails if any two compared n-bit blocks are equal.
  - If each call to a RNG produces fewer than 16 bits, the first n bits generated after power-up, initialization, or reset (for some n > 15) is saved for comparison with the next n generated bits. Each subsequent generation of n bits is compared with the previously generated n bits. The test fails if any two compared n-bit sequences are equal.

5.5.4 Severe Errors

When severe errors are detected (e.g. self-test failure or a conditional test failure) then all security related functions shall be disabled and no partial data is exposed through the data output interface. The only way to transition from the error state to an operational state is to reinitialize the cryptographic module (from an uninitialized state). The error state can be retrieved via the status interface (see paragraph 5.7.1 above). The following is the list of the API functions supported.

- ICC_GetStatus
- ICC_Init (CO)
- ICC_SetValue (CO)
- ICC_GetValue
- ICC_Attach (CO)
- ICC_Cleanup
- ICC_SelfTest
- ICC_GenerateRandomSeed
- ICC_MemCheck_start
- ICC_MemCheck_stop
- ICC_OBJ_nid2sn
- ICC_RAND_bytes
- ICC_RAND_seed
- ICC_EVP_PKEY_decrypt
- ICC_EVP_PKEY_encrypt
- ICC_EVP_PKEY_bits
- ICC_EVP_PKEY_size
- ICC_EVP_PKEY_new
- ICC_EVP_PKEY_free
- ICC_RSA_new

- ICC_EVP_get_digestbyname
- ICC_EVP_get_cipherbyname
- ICC_EVP_MD_CTX_new
- ICC_EVP_MD_CTX_free
- ICC_EVP_MD_CTX_init
- ICC_EVP_MD_CTX_cleanup
- ICC_EVP_MD_CTX_copy
- ICC_EVP_MD_type
- ICC_EVP_MD_size
- ICC_EVP_MD_block_size
- ICC_EVP_MD_CTX_md
- ICC_EVP_DigestInit
- ICC_EVP_DigestUpdate
- ICC_EVP_DigestFinal

- ICC_EVP_CIPHER_CTX_new
- ICC_EVP_CIPHER_CTX_free
- ICC_EVP_CIPHER_CTX_init
- ICC_EVP_CIPHER_CTX_cleanup
- ICC_EVP_CIPHER_CTX_set_key_length
- ICC_EVP_CIPHER_CTX_set_padding
- ICC_EVP_CIPHER_block_size
- ICC_EVP_CIPHER_key_length
- ICC_EVP_CIPHER_iv_length
- ICC_EVP_CIPHER_type
- ICC_EVP_CIPHER_CTX_cipher

- ICC_DES_random_key
- ICC_DES_set_odd_parity

- ICC_EVP_EncryptInit
- ICC_EVP_EncryptUpdate
- ICC_EVP_EncryptFinal
- ICC_EVP_DecryptInit
- ICC_EVP_DecryptUpdate
- ICC_EVP_DecryptFinal

- ICC_EVP_OpenInit
- ICC_EVP_OpenUpdate
- ICC_EVP_OpenFinal

- ICC_EVP_SealInit
- ICC_EVP_SealUpdate
- ICC_EVP_SealFinal

- ICC_RSA_generate_key
- ICC_RSA_check_key
- ICC_EVP_PKEY_set1_RSA
- ICC_EVP_PKEY_get1_RSA
- ICC_RSA_free

- ICC_RSA_private_encrypt
- ICC_RSA_private_decrypt
- ICC_RSA_public_encrypt
- ICC_RSA_public_decrypt

- ICC_i2d_RSAPrivateKey
- ICC_i2d_RSAPublicKey
- ICC_d2i_PrivateKey
- ICC_d2i_PublicKey

- ICC_EVP_PKEY_set1_DH
- ICC_EVP_PKEY_get1_DH
- ICC_DH_new
- ICC_DH_generate_key
- ICC_DH_check
- ICC_DH_compute_key
- ICC_DH_generate_parameters
- ICC_i2d_Dhparams
- ICC_d2i_Dhparams

- ICC_EVP_PKEY_set1_DSA
- ICC_EVP_PKEY_get1_DSA
- ICC_DSA_dup_DH
- ICC_DSA_sign
- ICC_DSA_verify
- ICC_DSA_new
- ICC_DSA_generate_key
- ICC_DSA_generate_parameters
- ICC_i2d_DSAPrivateKey
- ICC_d2i_DSAPrivateKet
- ICC_i2d_DSAPublicKey
- ICC_d2i_DSAPublicKey
- ICC_i2d_DSAparams
- ICC_d2i_DSAparams

- ICC_ERR_get_error
- ICC_ERR_peek_error
- ICC_ERR_peek_last_error
- ICC_ERR_error_string

- ICC_EVP_SignInit
- ICC_EVP_SignUpdate
- ICC_EVP_SignFinal
- ICC_EVP_VerifyInit
- ICC_EVP_VerifyUpdate
- ICC_EVP_VerifyFinal

- ICC_EVP_ENCODE_CTX_new
- ICC_EVP_ENCODE_CTX_free
- ICC_EVP_EncodeInit
- ICC_EVP_EncodeUpdate
- ICC_EVP_EncodeFinal
- ICC_EVP_DecodeInit
- ICC_EVP_DecodeUpdate
- ICC_EVP_DecodeFinal

- ICC_ERR_error_string_n
- ICC_ERR_lib_error_string
- ICC_ERR_func_error_string
- ICC_ERR_reason_error_string
- ICC_ERR_clear_error
- ICC_ERR_remove_state

The functions marked with (CO) are crypto officer functions

5.6 Design Assurance

The ICC module design team utilizes IBM's Configuration Management Version Control (CMVC) system.

CMVC integrates four facets of the software development process in a distributed development environment to facilitate project-wide coordination of development activities across all phases of the product development life cycle:

1. Configuration Management – the process of identifying, managing and controlling software modules as they change over time.
2. Version Control – the storage of multiple versions of a single file along with information about each version.
3. Change Control – centralizes the storage of files and controls changes to files through the process of checking files in and out.
4. Problem Tracking – the process of effectively tracking all reported defects and proposed design changes through to their resolution and implementation.

Files are stored in a file system on the server by means of a version control system. All other development data is stored in a relational database on the CMVC server. A CMVC client is a workstation that runs the CMVC client software (or browser for the web interface) to access the information and files stored on a CMVC server.
CMVC is used o perform the following tasks:

1. Organizing Development Data
2. Configuring CMVC Processes
3. Reporting Problems and Design Changes

4. Tracking Features and Defects

All source code is tracked using CMVC; documents are available in Lotus Notes database “Team Rooms” with version numbers assigned by document owner.

CMVC monitors changes with defects, features, and integrated problem tracking. Each of these restricts file changes so that they are made in a systematic manner. CMVC can require users to analyze the time and resources required to make changes, verify changes, and select files to be changed, approve work to be done, and test the changes. The requirements for changes are controlled by processes. Family administrators can create processes for components and releases to use, configuring them from CMVC sub processes.

Finally, the CMVC administrator policy mandates a regular audit of access check of all user accounts.

5.7 Mitigation Of Other Attacks

The cryptographic module is not designed to mitigate any specific attacks.

6. API Functions

The module API functions are fully described in the *IBM Crypto for C (ICC) Design Document.*