ActivIdentity Cryptographic Module for F5 and C5 Security Policy

This document is a non-proprietary security policy for ActivIdentity, Inc. Cryptographic Module for F5 and C5 (Cryptographic Module for F5 and C5) security software (also known as RSA BSAFE Crypto-Kernel 1.7.0.4).

Contents:

1 Preface

This document is a non-proprietary security policy for the Cryptographic Module for F5 and C5 cryptographic toolkit from ActivIdentity. This security policy describes how the Cryptographic Module for F5 and C5 toolkit meets the security requirements of FIPS 140-2, and how to securely operate it. This policy is prepared as part of the Level 1 FIPS 140-2 validation of the Cryptographic Module for F5 and C5 toolkit.

FIPS 140-2 (Federal Information Processing Standards Publication 140-2 - Security Requirements for Cryptographic Modules) details the U.S. Government requirements for cryptographic modules. More information about the FIPS 140-2 standard and validation program is available on the NIST website.

1.1 References

This document deals only with operations and capabilities of the Cryptographic Module for F5 and C5 in the technical terms of a FIPS 140-2 cryptographic toolkit security policy. More information about Cryptographic Module for F5 and C5 is available at http://www.actividentity.com/.

1.2 Document Organization

This Security Policy document is one document in the FIPS 140-2 Validation Submission package. With the exception of the Non-Proprietary *Cryptographic Module for F5 and C5 Security Policy*, the *FIPS 140-2 Validation Submission Documentation* is ActivIdentity-proprietary and is releasable only under appropriate non-disclosure agreements. For access to the documentation, please contact ActivIdentity.

This document explains the Cryptographic Module for F5 and C5 features and functionality relevant to FIPS 140-2, and contains the following sections:

- This section, “Preface” on page 2 provides an overview and introduction to the Security Policy.
- “Cryptographic Module for F5 and C5 Module” on page 3, describes Cryptographic Module for F5 and C5 and how it meets the FIPS 140-2 requirements.
- “Secure Operation of Cryptographic Module for F5 and C5” on page 10, addresses the required configuration for the FIPS140-mode of operation.
- “Acronyms” on page 16, lists the definitions for the acronyms used in this document.

2 Cryptographic Module for F5 and C5 Module

Cryptographic Module for F5 and C5 is a software development toolkit that offers optimized cryptographic algorithm implementations. Cryptographic Module for F5 and C5 provides the cryptographic foundation for ActivIdentity security products designed for C/C++ developers. Cryptographic Module for F5 and C5 is designed to offer the lowest level cryptographic application programming interfaces.

The features of this release of Cryptographic Module for F5 and C5 include:

- Support for the following cryptographic algorithms:
  - AES (128, 192, and 256-bit key sizes) in ECB and CBC.
  - ECDSA (NIST P-256 and P-384 curves).
  - HMAC-SHA-1, HMAC-SHA-256 and HMAC-SHA-384 (128 to 2048-bit key sizes).
  - RSA (1024, 2048, 3072 and 4096-bit key size).
  - SHA-1, SHA-256 and SHA-384.
  - Triple-DES (2 and 3 key) in ECB and CBC mode.
  - FIPS 186-2 Pseudo-Random Number Generator (PRNG) - Change Notice 1, without the mod q step.
- FIPS 140-2-validated cryptography.
- The binary produced for this release of the Cryptographic Module for F5 and C5 is a shared library targeted at systems running a Technologic Systems® TS-Linux operating system.

2.1 Cryptographic Module

Cryptographic Module for F5 and C5 is classified as a multi-chip standalone cryptographic module for the purposes of FIPS 140-2. As such, Cryptographic Module for F5 and C5 must be tested on a specific operating system and computer platform. Cryptographic Module for F5 and C5 was validated as meeting all FIPS 140-2 Level 1 security requirements, including cryptographic key management and operating system requirements.

Cryptographic Module for F5 and C5 is packaged as a set of dynamically loaded modules or shared library files that contain the module's entire executable code. The Cryptographic Module for F5 and C5 toolkit relies on the physical security provided by the host device in which it runs.

For FIPS 140-2 validation, Cryptographic Module for F5 and C5 is tested on the Technologic Systems TS-Linux 2.4.26-ts11, ARM920T (32-bit) platform.

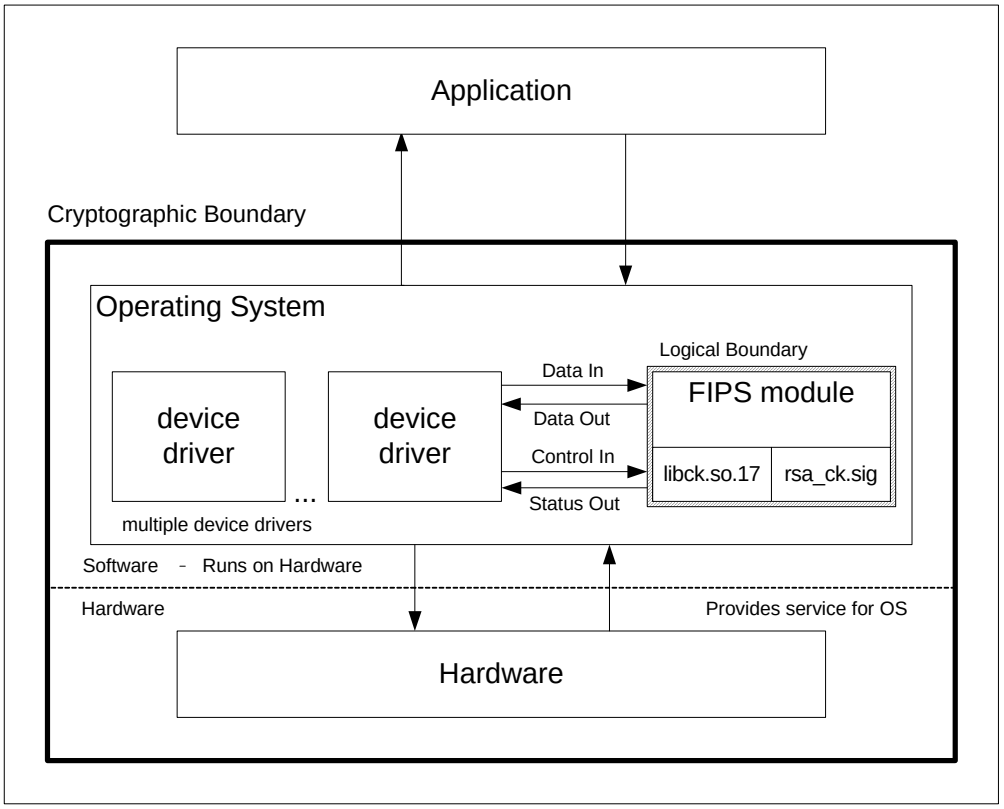
Note: Compliance is maintained on all of the above platforms for which the binary executable remains unchanged.

2.2 Cryptographic Module for F5 and C5 Interfaces

Cryptographic Module for F5 and C5 is evaluated as a multi-chip, standalone module. The physical interfaces for Cryptographic Module for F5 and C5 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 libck.so.1.7 and rsa_ck.sig. The underlying logical interface to Cryptographic Module for F5 and C5 is the application programming interface (API), which is documented in the *RSA BSAFE Crypto-Kernel Developer's Guide*. Cryptographic Module for F5 and C5 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 Cryptographic Module for F5 and C5 logical interfaces.

Figure 1 Cryptographic Module for F5 and C5 Logical Diagram



2.3 Roles and Services

Cryptographic Module for F5 and C5 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, Cryptographic Module for F5 and C5 does not support user identification or authentication for these roles. Only one role can be active at a time and Cryptographic Module for F5 and C5 does not allow concurrent operators.

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

Table 1 Cryptographic Module for F5 and C5 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 Crypto Officer Role

An operator assuming the Officer role can call any Cryptographic Module for F5 and C5 function. For the complete list of Cryptographic Module for F5 and C5 functions, see “Services” on page 12.

2.3.2 Crypto User Role

An operator assuming the User role can call any Cryptographic Module for F5 and C5 function except for `CK_FIPS_run_self_tests()`, which is reserved for the Officer. For the complete list of Cryptographic Module for F5 and C5 functions, see “Services” on page 12.

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

Cryptographic Module for F5 and C5 does not support key generation in this version of the module.

2.4.2 Key Storage

Cryptographic Module for F5 and C5 does not provide long-term cryptographic key storage. Cryptographic Module for F5 and C5 stores all cryptographic keys in volatile (short term) memory in plain text.

The following table lists all keys and Critical Security Parameters (CSPs) in the module and where they are stored.

Table 2 Key and CSP Storage

Key or CSP	Storage
Hardcoded HMAC key	Persistent storage embedded in the module binary (plaintext).
AES keys	Volatile memory only (plaintext).
Triple-DES keys	Volatile memory only (plaintext).
HMAC with SHA-1 and SHA-2 keys (SHA-256 and SHA-384)	Volatile memory only (plaintext).
ECDSA public/private keys	Volatile memory only (plaintext).
RSA public/private keys	Volatile memory only (plaintext).
FIPS 186-2 seed	Volatile memory only (plaintext).
FIPS 186-2 key	Volatile memory only (plaintext).

2.4.3 Key Access

An authorized operator of the module has access to all key data during Cryptographic Module for F5 and C5 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 3 Key and CSP Access

Service	Key or CSP	Type of Access
Encryption and decryption	AES keys	Read/Execute
Encryption and decryption	Triple-DES keys	Read/Execute
Hashing	None	N/A
MAC	HMAC keys	Read/Execute

Table 3 Key and CSP Access (continued)

Service	Key or CSP	Type of Access
Self-test (Crypto Officer services)	Hardcoded keys (HMAC)	Read/Execute
Show status	None	N/A
Sign and verify	ECDSA keys	Read/Execute
Sign and verify	RSA keys	Read/Execute
Zeroization	All	Read/Write

2.4.4 Key Zeroization

Cryptographic Module for F5 and C5 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.

Cryptographic Module for F5 and C5 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, digest, or asymmetric key 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 Cryptographic Module for F5 and C5, all of which are FIPS 140-2-approved.

Table 4 Cryptographic Module for F5 and C5 FIPS 140-2-approved Algorithms

Algorithm	Validation Certificate
AES in ECB and CBC modes (128, 192, and 256-bit).	1494
ECDSA (NIST P-256 and P-384 curves)	186
FIPS 186-2 Pseudo-Random Number Generator (PRNG) - Change Notice 1, without the mod q step.	813
HMAC-SHA-1, HMAC-SHA-256 and HMAC-SHA-384(0 to 2048 bit key sizes)	879
RSA (1024, 2048, 307,2 and 4096 bit key sizes)	733
SHA-1, SHA-256, and SHA-384	1347
Triple-DES (2 and 3 key) in ECB and CBC mode	1001

For more information about using Cryptographic Module for F5 and C5 in a FIPS 140-2-compliant manner, see “Secure Operation of Cryptographic Module for F5 and C5” on page 10.

2.6 Module Startup

At startup, Cryptographic Module for F5 and C5 loads the environment variables `RSA_CK_LIBRARY_PATH` and `RSA_CK_LIBRARY_TEST` and performs the module self-tests.

Note: If the `RSA_CK_LIBRARY_TEST` is set to “`TEST_MODE`”, the module loads the environment variable `RSA_CK_LIBRARY_TEST_FAIL_CASE`. If this is set (for example, ‘AES-192 ECB decrypt’) it invalidates the respective method pointer. The fail case values are stored in `ck_fips_140_module_params.h`

If the environment variable `RSA_CK_LIBRARY_PATH` fails to load 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 10.

2.6.1 Startup Self-tests

Cryptographic Module for F5 and C5 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, SHA-256, SHA-384 KATs
- HMAC-SHA1, HMAC-SHA256, and HMAC-SHA384 KATs
- ECDSA (NIST P-384 curve)
- FIPS 186-2 PRNG KAT
- RSA (1024 and 2048-bit key size)
- Triple-DES (2 and 3 key) in ECB and CBC mode
- RSA 2048-SHA256 and RSA 2048-SHA384 signature verification KATs.

Startup self-tests are executed automatically when Cryptographic Module for F5 and C5 is loaded into memory.

2.6.2 Mitigation of Other Attacks

RSA key operations implement blinding, a reversible way of modifying the input data so as to make the RSA operation immune to timing attacks. Blinding has no effect on the algorithm other than to mitigate attacks on the algorithm. 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.

3 Secure Operation of Cryptographic Module for F5 and C5

This section provides an overview of how to securely operate Cryptographic Module for F5 and C5 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 4, “Cryptographic Module for F5 and C5 FIPS 140-2-approved Algorithms,” on page 8, Cryptographic Module for F5 and C5 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 2048 bits.

3.2 Roles

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

Table 5 Cryptographic Module for F5 and C5 Roles

Role Identifier	Description
`CK_FIPS_ROLE_USER`	An operator assuming the User role can call any Cryptographic Module for F5 and C5 function except for `CK_FIPS_run_self_tests()`, which is reserved for the Officer. The complete list of Cryptographic Module for F5 and C5 functions is outlined in “Services” on page 12.
`CK_FIPS_ROLE_OFFICER`	An operator assuming the Officer role can call any Cryptographic Module for F5 and C5 function. The complete list of Cryptographic Module for F5 and C5 functions is outlined in “Services” on page 12.

3.3 Modes of Operation

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

Table 6 Cryptographic Module for F5 and C5 Modes of Operation

Mode	Description
`CK_FIPS_140_MODE_FIPS`	FIPS 140-2-approved. Enables the FIPS 140-2-approved algorithms listed in Table 4, “Cryptographic Module for F5 and C5 FIPS 140-2-approved Algorithms,” on page 8. This is the Cryptographic Module for F5 and C5 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 Cryptographic Module for F5 and C5

Module parameters are supplied to indicate the location of the module binary and the module signature data.

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.

Cryptographic Module for F5 and C5 can only be changed to `CK_FIPS_140_MODE_FIPS_TEST` mode if the module was initialized properly. The environment variable `RSA_CK_LIBRARY_TEST` 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 environment variable `RSA_CK_LIBRARY_TEST` is not set to "`TEST_MODE`". 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 environment variable `RSA_CK_LIBRARY_TEST` was set to "`TEST_MODE`" on startup of the module. Cryptographic Module for F5 and C5 can only be changed to `CK_FIPS_140_MODE_FIPS_TEST` mode if the module was initialized with the environment variable properly set to "`TEST_MODE`", 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 environment variable `RSA_CK_LIBRARY_TEST` set to "`TEST_MODE`".

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 Cryptographic Module for F5 and C5 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.

4 Services

The following table lists and describes the functions provided by Cryptographic Module for F5 and C5, and the key or CSP they access. For more information about these functions, see the *RSA BSAFE Crypto-Kernel Developer's Guide.*

Table 7 Cryptographic Module for F5 and C5 Services

Function	Description	Key or CSP
`CK_FIPS_MODULE_get_info()`	Retrieve product information about the Cryptographic Module for F5 and C5 FIPS 140-2 module.	None
`CK_FIPS_MODULE_get_status()`	Retrieve validity status information about the Cryptographic Module for F5 and C5 FIPS 140-2 module.	None
`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.	All
`CK_FIPS_free()`	Destroys the FIPS management context structure.	All
`CK_FIPS_get_mode()`	Returns the current mode of operation.	None
`CK_FIPS_set_mode()`	Sets a new mode for the operator.	None
`CK_FIPS_get_role()`	Returns the current role of the operator.	None
`CK_FIPS_set_role()`	Sets a new role for the operator.	None
`CK_FIPS_run_self_tests()`	Performs the Cryptographic Module for F5 and C5 power-up self-tests. The FIPS management context must be in the Officer role to successfully run the self-tests.	Hard coded keys (HMAC)
`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.	AES/Triple DES keys
`CK_FIPS_CIPHER_free()`	Finalizes the data of the cipher context structure and then frees any allocated memory.	AES/Triple DES keys
`CK_FIPS_CIPHER_params_set()`	Sets key and initialization vector (IV) data against the cipher context structure.	AES/Triple DES keys
`CK_FIPS_CIPHER_process()`	Performs the encryption or decryption action.	AES/Triple DES keys

Table 7 Cryptographic Module for F5 and C5 Services (continued)

Function	Description	Key or CSP
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.	HMAC keys
CK_FIPS_DIGEST_free()	Destroys the digest context.	HMAC keys
CK_FIPS_DIGEST_init()	Initializes the digest operation state before any data is added to it.	HMAC keys
CK_FIPS_DIGEST_update()	Processes a buffer of input data and updates the state of the digest or MAC operation.	HMAC keys
CK_FIPS_DIGEST_final()	Completes the digest or MAC operation and returns the digest result.	HMAC keys
CK_FIPS_DIGEST_params_set()	Sets the key data for a MAC operation.	HMAC keys
CK_FIPS_DIGEST_mac_digest_set()	Sets the digest method table that is used for a MAC operation.	HMAC keys
CK_FIPS_DIGEST_length()	Returns the length of the digest.	HMAC keys
CK_FIPS_SIGN_new()	Creates a new sign context for signature operations according to the FIPS control state held by the FIPS management context.	RSA/ECDSA keys
CK_FIPS_SIGN_free()	Finalizes the data of the sign context structure then frees any allocated memory.	RSA/ECDSA keys
CK_FIPS_SIGN_params_set()	Sets RSA private key and modules or ECDSA private key and curve type against the sign context structure.	RSA/ECDSA keys
CK_FIPS_SIGN_process()	Performs the signature operation.	RSA/ECDSA keys
CK_FIPS_SIGN_set_salt_len()	Specifies the salt length for RSA PSS signature generation.	RSA/ECDSA keys
CK_FIPS_VERIFY_new()	Creates a new verify context for signature operations according to the FIPS control state held by the FIPS management context.	RSA/ECDSA keys
CK_FIPS_VERIFY_free()	Finalizes the data of the verify context structure then frees any allocated memory.	RSA/ECDSA keys
CK_FIPS_VERIFY_params_set()	Sets RSA public key and modules or ECDSA public key and curve type against the verify context structure.	RSA/ECDSA keys
CK_FIPS_VERIFY_process()	Performs the verification operation	RSA/ECDSA keys

Table 7 Cryptographic Module for F5 and C5 Services (continued)

Function	Description	Key or CSP
`R2_ALG_MFUNC_pkcs1_raw_sign()`	Returns the method chain for RSA PKCS #1 raw signing.	RSA keys
`R2_ALG_MFUNC_pkcs1_raw_sign_blinding()`	Returns the method chain for RSA PKCS #1 raw signing with blinding performed	RSA keys
`R2_ALG_MFUNC_pkcs1_raw_verify()`	Returns the method chain for RSA PKCS #1 raw verification,	RSA keys
`R2_ALG_MFUNC_pss_sign()`	Returns the method chain for RSA PSS signing.	RSA keys
`R2_ALG_MFUNC_pss_sign_blinding()`	Returns the method chain for RSA PSS signing with blinding performed.	RSA keys
`R2_ALG_MFUNC_pss_verify()`	Returns the method chain for RSA PSS raw verification.	RSA keys
`R2_ALG_MFUNC_rsa_raw_verify()`	Returns the method chain for RSA raw verification.	RSA keys
`R2_ALG_MFUNC_x931_sign()`	Returns the method chain for RSA X.931 signing.	RSA keys
`R2_ALG_MFUNC_x931_sign_blinding()`	Returns the method chain for RSA X.931 signing with blinding performed.	RSA keys
`R2_ALG_MFUNC_x931_verify()`	Returns the method chain for X.931 verification	RSA keys
`R2_ALG_MFUNC_ecdsa_sign()`	Returns the method chain for ECDSA signing.	ECDSA keys
`R2_ALG_MFUNC_ecdsa_verify()`	Returns the method chain for ECDSA verification.	ECDSA keys
`R2_ALG_MFUNC_pkcs1_asn1_sign()`	Returns the method chain for RSA ASN.1 signing.	RSA keys
`R2_ALG_MFUNC_pkcs1_asn1_sign_blinding()`	Returns the method chain for RSA ASN.1 signing with blinding performed.	RSA keys
`R2_ALG_MFUNC_pkcs1_asn1_verify()`	Returns the method chain for RSA ASN.1 verification.	RSA keys
`R1_CIPH_METH_aes_cbc_fast()`	Returns the method functions for AES encryption in CBC mode that are optimized for performance.	AES keys
`R1_CIPH_METH_aes_ecb_fast()`	Returns the method functions for AES encryption in ECB mode that are optimized for performance.	AES keys
`R1_CIPH_METH_des3_cbc_fast()`	Returns the method functions for Triple-DES encryption in CBC mode that are optimized for performance.	Triple DES keys

Table 7 Cryptographic Module for F5 and C5 Services (continued)

Function	Description	Key or CSP
`R1_CIPH_METH_des3_ecb_fast()`	Returns the method functions for Triple-DES encryption in ECB mode that are optimized for performance.	Triple DES keys
`R1_DGST_METH_sha1_fast()`	Returns the method function for SHA1 encryption.	None
`R1_DGST_METH_sha256_fast()`	Returns the method functions for SHA256 encryption.	None
`R1_DGST_METH_sha384_fast()`	Returns the method function for SHA384 encryption.	None
`R1_DGST_METH_hmac()`	Returns the method for HMAC.	HMAC keys

5 Acronyms

The following table lists the acronyms used with Cryptographic Module for F5 and C5 and their definitions.

Table 8 Acronyms used with Cryptographic Module for F5 and C5

Acronym	Definition
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.
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 Parameters.
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.
ECDSA	Elliptic Curve Digital Signature Algorithm.
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. Various types of keys include: distributed key, private key, public key, secret key, session key, shared key, subkey, symmetric key, and weak key.

Table 8 Acronyms used with Cryptographic Module for F5 and C5 (continued)

Acronym	Definition
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.
Privacy	The state or quality of being secluded from the view or presence of others.
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-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 symmetric encryption algorithm which uses either two or three DES keys. The two key variant of the algorithm provides 80 bits of security strength while the three key variant provides 112 bits of security strength.