# RSA BSAFE® Crypto CDC Module 1.1 Security Policy

This document is a non-proprietary security policy for RSA BSAFE Crypto CDC Module 1.1 security software. For the remainder of this document the RSA BSAFE Crypto CDC Module will be referred to as the Module.

## Contents:

# 1 Preface

This document is a non-proprietary security policy for the RSA BSAFE Crypto CDC Module. This Module is contained in the TLS-J ME security toolkit from RSA, the Security Division of EMC (RSA). This security policy describes how the Module 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 Module.

The Module provides the BSAFE Application Programming Interface (API) in the `cryptoCDCFIPS.jar` file.

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 from this website: `http://csrc.nist.gov/`.

## 1.1 References

This document deals only with operations and capabilities of the Module in the technical terms of a FIPS 140-2 cryptographic toolkit security policy. More information on TLS-J ME and the entire RSA BSAFE product line is available at:

- `http://www.rsa.com/` for information on the full line of products and services.
- `http://www.rsa.com/node.aspx?id=1319` for an overview of security tools for Java developers.
- `http://www.rsa.com/node.aspx?id=1204` for an overview of the RSA BSAFE product range.

## 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 *Crypto CDC Module 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 the documentation, please contact RSA Security.

This document explains the Module's 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.
- "The Cryptographic Toolkit" on page 4, describes the Module and how it meets the FIPS 140-2 requirements.
- "Secure Operation of the Module" on page 13, provides information on implementing the FIPS140 mode of operation.
- "Acronyms" on page 16, lists the definitions for the acronyms used in this document.

# 2 The Cryptographic Toolkit

This section provides an overview of the Module, and contains the following topics:

## 2.1 Introduction

More than a billion copies of the RSA BSAFE technology are embedded in today's most popular software applications and hardware devices. Encompassing the most widely-used and rich set of cryptographic algorithms as well as secure communications protocols, RSA BSAFE software is a set of complementary security products relied on by developers and manufacturers worldwide.

The TLS-J ME software library is a software development toolkit which enables developers to incorporate cryptographic, certificate, and Transport Layer Security (TLS) capabilities into their applications.

## 2.2 Toolkit Characteristics

The Module is classified as a FIPS 140-2 multi-chip standalone module. As such, the Module is tested on particular operating systems and computer platforms. The cryptographic boundary includes the Module running on selected platforms that are running selected operating systems.

The Module is validated for all FIPS 140-2 Level 1 security requirements. It is packaged in a Java Archive (JAR) file containing all the code for the toolkit. In addition, the Module relies on the physical security provided by the host on which it runs.

The Module is provided in the `cryptoCDCFIPS.jar` file.

The Module is tested on the following platform:

- Microsoft Windows XP SP3 (32-bit) , with the Sun Microsystems Java ME SDK 3.0 emulator runtime environment.

For this validation the module was not tested on the following platforms but RSA is allowed to claim compliance since RSA attests the source code remains unchanged based on CMVP Implementation Guidance (IG) G.5 (User). This includes (but is not limited to):

- Microsoft Windows XP SP3 (x86) with Java ME platform SDK 3.0 and Sun Emulator for CDC 1.1/Foundation Profile 1.1
- Microsoft Windows XP SP3 x86 with Java ME platform SDK 3.0 and Sun Emulator for CDC 1.1/Foundation Profile 1.1 with optional JCE provider package.
- Canon's imageRUNNER with MEAP SDK 3.6 and CDC 1.1 Foundation Profile 1.1 with optional JCE provider package.

For a resolution on the issue of multi-user modes, see the NIST document `Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program`.

## 2.3 Toolkit Interfaces

As a multi-chip standalone toolkit, the physical interface to the Module consists of a keyboard, mouse, monitor, serial ports and network adapters.

The underlying logical interface to the toolkit is the API, documented in the *RSA BSAFE TLS-J ME Javadoc*. 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 in the returns and error codes documented for each call. This is shown in the following diagram.

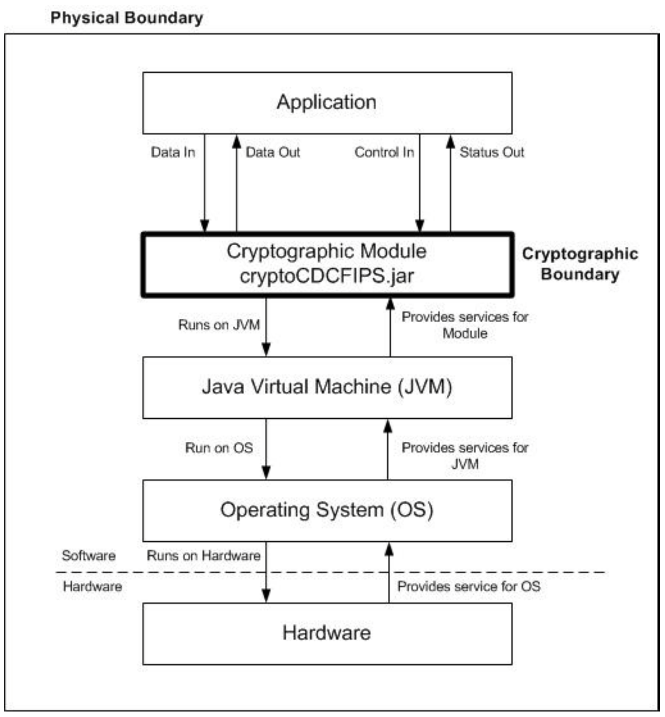

**Figure 1 Logical Diagram**

## 2.4 Roles and Services

The Module meets all FIPS140-2 Level 1 requirements for Roles and Services, implementing both a Crypto Officer role and a Crypto User role. As allowed by FIPS 140-2, the Module does not require user identification or authentication for these roles.

### 2.4.1 Crypto Officer Role

The Crypto Officer role is responsible for installation of the toolkit. An operator can assume the Crypto Officer role by instantiating the `CryptoProvider` class with `com.rsa.jme.FIPS140Context.OFFICER_FIPS140` or `com.rsa.jme.FIPS140Context.OFFICER_FIPS140_SSL` as a parameter.

The Crypto Officer is provided with all the services available to the Crypto User (see section 2.4.2). In addition, the Crypto Officer can explicitly re-execute the power-up self-tests after the toolkit has been loaded. This can be done using the `com.rsa.jme.CryptoModule.FIPS140.runSelfTests` method with `FIPS140Context.OFFICER_FIPS140` or `FIPS140Context.OFFICER_FIPS140_SSL` as the argument.

**Note:** When the Module is loaded and configured for FIPS140-2 use, the power-up self tests run automatically. If the `CryptoModule.FIPS140.runSelfTests` method is invoked after the toolkit is loaded, all power-up tests will be re-executed.

### 2.4.2 Crypto User Role

The Crypto User role is the default operating role. An operator can explicitly assume the Crypto User role by instantiating the `CryptoProvider` class with no parameters, or with `com.rsa.jme.FIPS140Context.USER_FIPS140` or `com.rsa.jme.FIPS140Context.USER_FIPS140_SSL` as a parameter.

### 2.4.3 Services

The following table lists the services and related classes provided by the Module in terms of the toolkit interface.

Table 1 Authorized Services and Related Classes

| Service | Related Classes |
|---|---|
| Encryption and decryption | `Cipher.java` |
| Digital signature and verification | `Signature.java` |
| Hashing | `MessageDigest.java` |
| MAC | `MAC.java` |
| Random number generation | `SecureRandom.java` |
| Key establishment primitives | `KeyAgreement.java` |
| Key generation | `KeyPairGenerator.java` |
| Self-test (only available to Crypto Officer service) | `CryptoModule.FIPS140.runSelfTests()` |
| Zeroization | `SensitiveData.java` |

**Note:** All the services except for the self-test method are available when in Crypto User Role.

## 2.5 Cryptographic Key Management

### 2.5.1 Key Generation

The Module supports the generation of the DSA, RSA, and Diffie-Hellman (DH) and ECC public and private keys. The toolkit also employs a Federal Information Processing Standard 186-2, Digital Signature Standard (FIPS 186-2) Approved Random Number Generator, a FIPS Approved HMAC Deterministic Random Bit Generator (HMAC DRBG SP800-90), as well as a FIPS Approved Dual Elliptic Curve Deterministic Random Bit Generator (ECDRBG SP 800-90) for generating asymmetric and symmetric keys used in algorithms such as AES, Triple-DES, RSA, DSA, DH and ECC.

### 2.5.2 Key Security

All key data resides in internally allocated data structures and can only be output using the Module's API. The operating system and the Java Runtime Environment (JRE) safeguard memory and process space from unauthorized access.

### 2.5.3 Key Access

An authorized operator of the Module has access to all key data created during the operation of the Module.

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

The following table lists the different services provided by the toolkit with the type of access to keys or Critical Security Parameters (CSPs).

Table 2 Key and CSP Access

| **Service** | **Key or CSP** | **Type of Access** |
|---|---|---|
| Encryption and decryption | Symmetric keys (AES, Triple-DES) | Read/Execute |
| Digital signature and verification | Asymmetric keys (DSA, RSA, ECDSA) | Read/Execute |
| Hashing | None | N/A |
| MAC | HMAC keys | Read/Execute |
| Random number generation | FIPS 186-2 seed and key<br>HMAC DRBG Entropy, V Value, Key, and init_seed<br>EC DRBG Entropy, S Value, and init_seed | Read/Write/Execute |
| Key establishment primitives | Asymmetric keys (RSA, DH, ECDH) | Read/Execute |

Table 2 Key and CSP Access

| Service | Key or CSP | Type of Access |
|---|---|---|
| Key generation | Symmetric keys (AES, Triple-DES)<br>Asymmetric keys (DSA, EC, RSA, DH)<br>MAC keys (HMAC) | Write |
| Self-test (only available to Crypto Officer service) | Hardcoded keys<br>(AES, Triple-DES, RSA, DSA, ECDSA and HMAC) | Read/Execute |
| Zeroization | All | Read/Write |

### 2.5.4 Key Zeroization

Users can ensure sensitive data is properly zeroized by making use of the `SensitiveData.clearSensitiveData` method for clearing sensitive data. The toolkit ensures that all ephemeral sensitive data is cleared within the toolkit.

### 2.5.5 Key Storage

The Module does not provide long-term cryptographic key storage. Storage of keys is the responsibility of the user of the Module.

The following table shows how the storage of keys and CSPs are handled. The Crypto User and Crypto Officer roles have equal and complete access to all keys and CSPs.

Table 3 Key and CSP 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) |
| EC public keys | In volatile memory only (plaintext) |
| EC private keys | In volatile memory only (plaintext) |
| DH public key | In volatile memory only (plaintext) |
| DH 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) |

Table 3 Key and CSP Storage

| | |
|---|---|
| PRNG Keys (FIPS 186-2) | In volatile memory only (plaintext) |
| EC DRBG Entropy | In volatile memory only (plaintext) |
| EC DRBG S Value | In volatile memory only (plaintext) |
| EC DRBG init_seed | In volatile memory only (plaintext) |
| HMAC DRBG Entropy | In volatile memory only (plaintext) |
| HMAC DRBG V Value | In volatile memory only (plaintext) |
| HMAC DRBG Key | In volatile memory only (plaintext) |
| HMAC DRBG init_seed | In volatile memory only (plaintext) |
| HMAC Integrity Test Key | In Module JAR file (plaintext) |

## 2.6 Cryptographic Algorithms

The Module meets FIPS 140-2 requirements by implementing algorithm enforcement, such that when operating in `FIPS140_MODE`, only FIPS 140-approved algorithms are available for use.

The following table lists the FIPS 140-approved algorithms provided by the Module, when operating in `FIPS140_MODE`.

Table 4 FIPS-approved Algorithms in the Module

| **Algorithm** | **Validation Certificate** |
|---|---|
| AES ECB, CBC, GCM | Certificate #1244 |
| Triple-DES ECB, CBC | Certificate #891 |
| Diffie-Hellman primitives | Non-Approved (Allowed in FIPS mode) |
| DSA | Certificate #410 |
| Dual EC DRBG (SP800-90) | Certificate #28 |
| EC-Diffie-Hellman primitives, EC-Diffie-Hellman with Cofactor primitives | Non-Approved (Allowed in FIPS mode) |
| EC-DSA, EC-DSA-SHA1 | Certificate #146 |
| FIPS 186-2 PRNG (Change Notice 1 without the mod q step) | Certificate #691 |
| HMAC DRBG (SP800-90) | Certificate #28 |
| HMAC-SHA1, SHA224, SHA256, SHA384, SHA512 | Certificate #727 |

Table 4 FIPS-approved Algorithms in the Module

| Algorithm | Validation Certificate |
|---|---|
| RSA encrypt/decrypt | Non-Approved (Allowed in FIPS mode for key transport) |
| RSA X9.31, PKCS #1 V.1.5, PKC S#1 V.2.1 | Certificate #597 |
| SHA-1 | Certificate #1143 |
| SHA-224, 256, 384, 512 | Certificate #1143 |

The following list contains the non-FIPS 140-approved algorithms provided by the Module, when operating in `NON_FIPS140_MODE`.

- DES
- ECIES
- MD4
- MD5
- PBE
- RC2® block cipher
- RC4® stream cipher
- RSA OAEP for key transport
- Raw RSA encryption and decryption
- HMAC-MD5.

## 2.7 Self-tests

The Module performs power-up and conditional self-tests to ensure proper operation. If the power-up self-test fails, the toolkit is disabled and throws a `SecurityException`. The toolkit can only leave the disabled state by restarting the JVM. If the conditional self-test fails, the toolkit throws a `SecurityException` and aborts the operation. A conditional self test failure does NOT disable the toolkit.

### 2.7.1 Power-up Self-tests

The following power-up self-tests are implemented in The Module:

- FIPS186 PRNG KAT
- AES KAT
- TDES KAT
- SHA-1 KAT
- SHA-224 KAT

- SHA-256 KAT
- SHA-384 KAT
- SHA-512 KAT
- MD5 KAT
- HMAC SHA-1 KAT
- HMAC SHA-224 KAT
- HMAC SHA-256 KAT
- HMAC SHA-384 KAT
- HMAC SHA-512 KAT
- HMAC DRBG Self-Test
- ECDRBG Self-Test
- ECDSA KAT
- DSA KAT
- DSA, RSA, EC pair-wise consistency test
- RSA (signature) KAT
- Software/firmware integrity check.

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

### 2.7.2 Conditional Self-tests

The Module performs two conditional self-tests:

- Pair-wise consistency tests each time the toolkit generates a DSA, RSA or EC public/private key pair.
- Continuous RNG (CRNG) test each time the toolkit produces random data, as per the FIPS 186-2 standard. The CRNG test is performed on all approved RNGs.

### 2.7.3 Mitigation of Other Attacks

RSA key operations implement blinding. Blinding is 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 values are squared for each operation.

# 3 Secure Operation of the Module

The Module does not require any special configuration to operate in conformance with FIPS 140-2 requirements. The following guidance must be followed, however, to achieve a FIPS140 mode of operation.

## 3.1 Crypto User Guidance

The Crypto User must only use algorithms approved for use in a FIPS 140 mode of operation, as listed in Table 4, "FIPS-approved Algorithms in the Module," on page 10. The requirements for using the approved algorithms in a FIPS 140 mode of operation are as follows:

- The bit-length for a DSA key pair must be 1024 bits.
- Random Number Generators must be seeded with values of at least 160 bits in length.
- Bit lengths for an RSA[1] key pair must be 1024, 2048 or 3072.
- Bit lengths for the Diffie-Hellman[2] key agreement must be 1024 or 2048 bits. Diffie-Hellman shared secret provides between 80 bits and 112 bits of encryption strength.
- Bit lengths for an HMAC key must be one half of the block size.
- If RSA key generation is requested in FIPS140 mode, the toolkit always uses the FIPS140-approved RSA X9.31 key-generation procedure. Key wrapping methodology provides between 80 and 128 bits of encryption strength.
- When using an Approved RNG to generate keys, the RNG's requested security strength must be at least as great as the security strength of the key being generated. For more information on requesting the RNG security strength, see the JCE API and BSAFE API Random Number Generation sections of the *RSA BSAFE TLS-J ME 1.2 Developers Guide*.

More information on the algorithm strength and keysize is provided in the *RSA BSAFE TLS-J ME 1.1 Release Notes*.

Users should take care to zeroize CSPs when they are no longer needed.

## 3.2 Crypto Officer Guidance

The Crypto Officer is responsible for installing the toolkit. Installation instructions are provided in the *4A-TLS-J ME Installation Guide*.

When operating the toolkit after installation, the Crypto Officer must follow the Crypto User guidance requirements detailed in section 3.1.

[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.

## 3.3 Operating the Cryptographic Module

The Module operates in `FIPS140_MODE` by default. When using the Module in this FIPS approved mode, the Module ensures that only the FIPS approved algorithms listed in "Services" on page 7 are available to operators.

The Service `CryptoModule.FIPS140.runSelfTests()` is restricted to operation by the Crypto Officer.

## 3.4 Modes of Operation

There are three modes of operation:

- `FIPS140_MODE`
- `FIPS140_SSL_MODE`
- `NON_FIPS140_MODE.`

The following table lists the available modes, and the algorithms available in those modes. Cryptographic algorithms can be created in different modes using the associated `com.rsa.jme.FIPS140Context` instance to instantiate a `CryptoProvider` object. For more information about operating in FIPS 140 modes, see the *RSA BSAFE TLS-J ME Javadoc*.

Table 5 Mode Value to Change the Mode of Operation

| Mode Value | Algorithms Available |
|---|---|
| `FIPS140Context.MODE_FIPS140` FIPS 140-2 approved. | Provides the cryptographic algorithms listed inTable 4, "FIPS-approved Algorithms in the Module," on page 10. This is the default mode on start up. |
| `FIPS140Context.MODE_FIPS140_SSL` FIPS 140-2 approved if used with TLS protocol implementations. | Provides the same algorithms as `FIPS140Context.MODE_FIPS140`, plus the MD5 message digest.<br>This mode can be used in the context of the key establishment phase in the TLSv1, TLSv1.1 and TLSv1.2 protocols. For more information, see section 7.1 Acceptable Key Establishment Protocols in `Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program`.<br>The implementation guidance disallows the use of the SSLv2 and SSLv3 versions. Cipher suites that include non-FIPS 140-2-approved algorithms are unavailable.<br>This mode allows implementations of the TLS protocol to operate the Module in a FIPS 140-2-compliant manner. |
| `FIPS140Context.MODE_NON_FIPS140` Not FIPS 140-2 approved. | Allows users to operate the Module without any cryptographic algorithm restrictions. |

## 3.5 Startup Self Tests

All KATs are executed on toolkit start-up, which occurs on first use. If any KAT fails, the toolkit is disabled.

## 3.6 Default Random Number Generator

The Module provides a default RNG, which is HMAC-DRBG, with 128-bit security, using SHA-256.

# 4 Acronyms

The following table lists the acronyms used in this document and their definitions.

Table 6 Acronyms and Definitions

| Acronym | Definition |
|---|---|
| 3DES | Refer to Triple-DES |
| 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. Attack types include an algebraic attack, birthday attack, brute force attack, chosen ciphertext attack, chosen plaintext attack, differential cryptanalysis, known plaintext attack, linear cryptanalysis, middleperson attack and timing 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. |
| CRNG | Continuous Random Number Generation. |
| CSP | Critical Security Parameters. |
| DES | Data Encryption Standard. A symmetric encryption algorithm with a 56-bit key. |
| 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. |
| DRBG | Deterministic Random Bit Generator. |
| DSA | Digital Signature Algorithm. An asymmetric algorithm for creating digital signatures. |
| EC | Elliptic Curve. |
| ECB | Electronic Code Book. A mode of encryption in which identical plaintexts are encrypted to identical ciphertexts, given the same key. |
| ECC | Elliptic Curve Cryptography. |
| ECDH | Elliptic Curve Diffie-Hellman. |

Table 6 Acronyms and Definitions

| Acronym | Definition |
|---|---|
| ECDHC | Elliptic Curve Diffie-Hellman with Cofactor. |
| ECDSA | Elliptic Curve Digital Signature Algorithm. |
| ECIES | Elliptic Curve Integrated Encryption Scheme. |
| 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. |
| IV | Initialization Vector.<br>Used as a seed value for an encryption operation. |
| JVM | Java Virtual Machine. |
| 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. |
| MD4 | A message digest algorithm which implements a cryptographic hash function, created by Rivest. |
| MD5 | A message digest algorithm which implements a cryptographic hash function with a 128-bit hash value, created by Rivest. |
| 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. |
| 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. |

Table 6 Acronyms and Definitions

| Acronym | Definition |
|---|---|
| RC4 | Symmetric algorithm designed by Ron Rivest using variable length keys (usually 40 bit or 128 bit). |
| 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 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. |
| TDES | Refer to Triple-DES |
| 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. |