TrustConnector 2 v2.0 with StrongClient v4.0; TrustConnector 2 v2.0 with StrongClient v4.0 and StrongROM v3.1 Security Policy

Document Version 2.7

Phoenix Technologies

July 14, 2006

TABLE OF CONTENTS

1. Module Overview

The Phoenix Technologies TrustConnector 2 v2.0 with StrongClient v4.0; TrustConnector 2 v2.0 with StrongClient v4.0 and StrongROM v3.1 (hereafter referred to as TrustConnector 2) product is a FIPS-140-2 Level 1 compliant software module that implements a standard Cryptographic Service Provider (CSP) for Microsoft CryptoAPI. TrustConnector 2 is built to transparently support PKI-aware Windows applications that support Microsoft CryptoAPI, certificate-aware network infrastructure products, and certificate-based enterprise network applications and management products. It enables built-in device authentication and transparently enhances the way Windows protects identity credentials associated with digital certificates. It binds the credentials to the platform to which they are issued by using a high entropy key that is bound to the hardware the application is running on.

TrustConnector 2 consists of the following components: Microsoft CryptoAPI, TrustConnector CSP (version 2.0), TrustConnector StrongClient Software Crypto Engine (version 4.0), an optional TrustedCore StrongROM Firmware Crypto Engine (version 3.1), and the embedded FIPS-140-1 validated Microsoft Enhanced Cryptographic Provider (RSAENH). These components comprise the module's logical boundary. The TrustConnector CSP may be configured to use a Trust Platform Module (TPM). The TPM is outside of the module logical boundary and may not be used when running in FIPS mode. The physical cryptographic boundary of the module is defined as the enclosure of the computer system on which the cryptographic module is to be executed.

TrustConnector 2 may be configured in one of two configurations.

Configuration 1	TrustConnector 2 with StrongClient and StrongROM (SW Version TrustConnector 2 v2.0, StrongClient v4.0; FW Version StrongROM v3.1)
	A General Purpose Computer with the TrustConnector Cryptographic Service Provider, TrustConnector StrongClient Software Crypto Engine, TrustedCore StrongROM Firmware Crypto Engine.
	Optionally, Firmware Crypto engine can have access to an Intel Hardware Random Number Generator (FWH80802).
Configuration 2	TrustConnector 2 with StrongClient (without StrongROM) (SW Version TrustConnector 2 v2.0, StrongClient v4.0)
	A General Purpose Computer with only the TrustConnector Cryptographic Service Provider and TrustConnector StrongClient Software Crypto Engine.

For all configurations, the physical embodiment of the module, as defined in FIPS-140-2, is Multi-Chip Standalone.

There are three cryptographic engines in TrustConnector 2.

1) TrustedCore StrongROM Firmware Crypto Engine: This engine is the preferred cryptographic engine for RSA encryption and decryption (for key transport and digital signatures), AES and HMAC-SHA-1 when used to protect RSA Private Keys, and random number generation for creating new AES keys and new RSA keys.

2) TrustConnector StrongClient Software Crypto Engine: This cryptographic engine is used if the optional TrustedCore StrongROM Firmware Crypto Engine is not available. This engine performs the same cryptographic operations the TrustedCore StrongROM Firmware Crypto Engine would if it were present.

3) Microsoft Enhanced Cryptographic Provider: This cryptographic engine is used for all cryptographic operations not done by either the TrustedCore StrongROM Firmware Crypto Engine or the TrustConnector StrongClient Software Crypto Engine. The Microsoft Enhanced Cryptographic Provider is an embedded FIPS-140-1 validated module. There have been no changes made to this embedded module. The certificate number for the Microsoft Enhanced Cryptographic Provider is certificate number 238.

An overview of the module's, interfaces and boundaries appears in Figure 1.

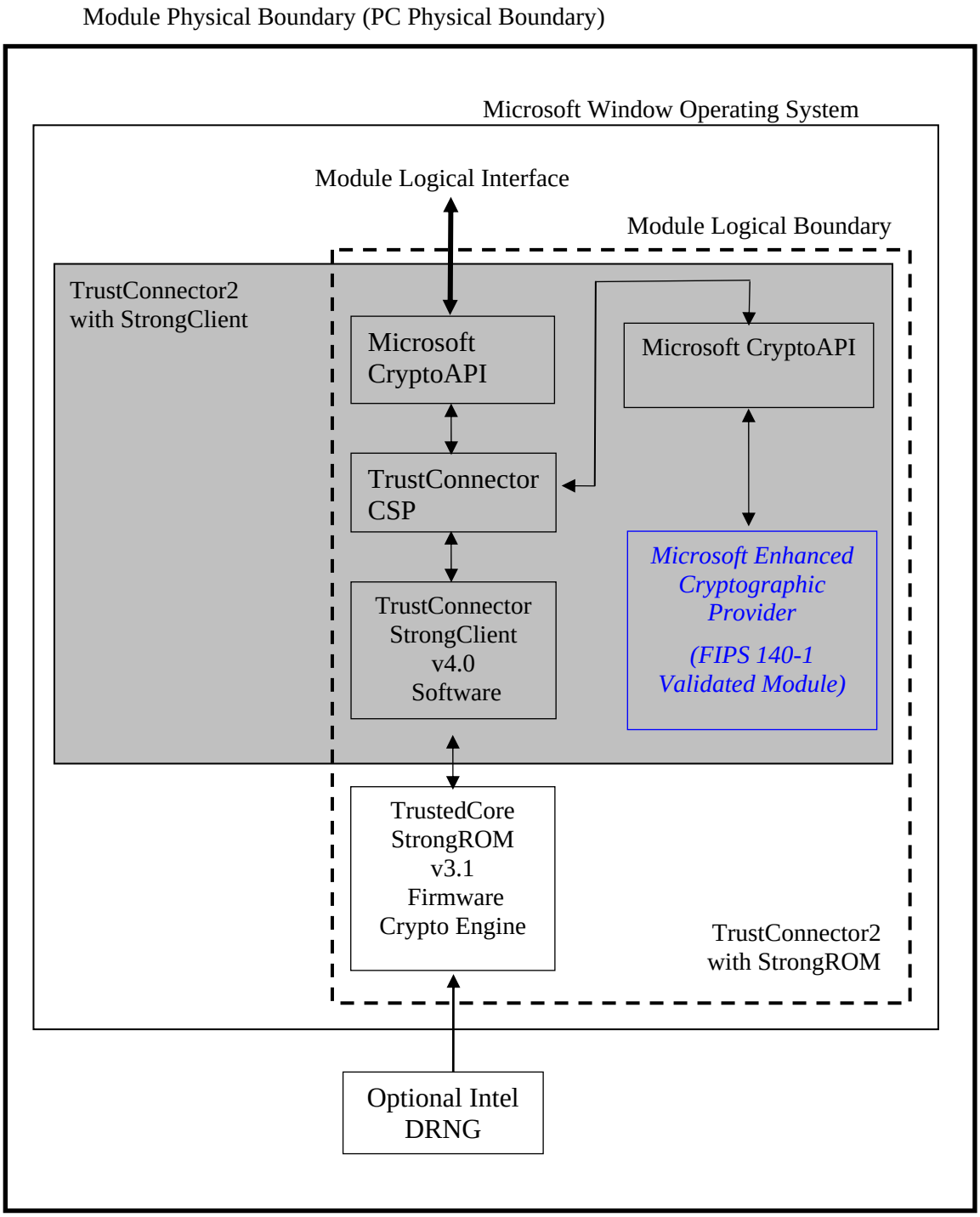

Figure 1: TrustConnector 2 Components, Interfaces and Boundary

2. Security Level

The cryptographic module meets the overall requirements applicable to Level 1 security of FIPS-140-2.

Table 1 - Module Security Level Specification

Security Requirements Section	Level
Cryptographic Module Specification	1
Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	1*
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	3
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

* Physical Security is not applicable for Configuration 2, TrustConnector 2 with StrongClient (without StrongROM)

3. Modes of Operation

The module supports both approved and non-approved modes of operation.

Approved mode of operation

In FIPS mode, the TrustConnector 2 supports FIPS approved algorithms as follows:

TrustConnector 2 with StrongROM

• RSA	Certificate Number 114
• AES	Certificate Number 343
• HMAC-SHA-1	Certificate Number 83, Vendor affirmed to be FIPS 198 compliant; Certificate Number 105
• SHA-1	Certificate Numbers 83, 418
• Triple-DES	Certificate Number 81
• DES (transitional phase only – valid until May 19, 2007) ; DES MAC (transitional phase only - valid until May 19, 2007)	Certificate Number 156
• DRNG – FIPS 186-2 (+Change Notice) Appendix 3.1 based on SHA-1 X-original for general purpose values	Certificate Number 118

TrustConnector 2 with StrongClient

• RSA	Certificate Number 115
• AES	Certificate Number 344
• HMAC-SHA-1	Certificate Number 83, Vendor affirmed to be FIPS 198 compliant; Certificate Number 147
• SHA-1	Certificate Numbers 83, 419
• Triple-DES	Certificate Number 81
• DES (transitional phase only – valid until May 19, 2007) ; DES MAC (transitional phase only - valid until May 19, 2007)	Certificate Number 156
• DRNG – FIPS 186-2 (+Change Notice) Appendix 3.1 based on SHA-1 X-original for general purpose values	Certificate Number 164

DES is designed to be used by legacy systems, such as legacy e-mail systems, that have been designed to use DES via the Microsoft Crypto API.

The embedded Microsoft Enhanced Cryptographic Provider, a FIPS 140-1 validated module, uses an approved deterministic algorithm to generate random numbers. The algorithm used is from FIPS 186-2 appendix 3.1 with SHA-1 as the G function.

In FIPS mode, the TrustConnector 2 supports the following non-approved algorithms:

- RSA Key Transport
- Nondeterministic RNG to provide seeds to the FIPS approved Firmware DRNG
- Nondeterministic RNG to provide seeds to the FIPS approved Software DRNG
- Non-deterministic Hardware RNG (optional)

RSA Key Transport is available in FIPS mode because it meets all of the FIPS 140-2 Annex D criteria for asymmetric key establishment techniques. Note that RSA Key Transport provides 80 bits of strength using a 1024 bit RSA key and 112 bits of strength using a 2048 bit RSA key. The two non-approved Nondeterministic RNG algorithms are used only for seeding the FIPS approved DRNG algorithms.

To operate TrustConnector 2 in the FIPS approved mode operators must:

- Run TrustConnector 2 on Microsoft Windows XP Professional Service Pack 2 operating system in single user mode
- Use only FIPS approved algorithms or algorithms approved for use in FIPS mode
- Use only FIPS approved Microsoft CryptoAPI function calls (see the list in Non-FIPS mode of operation below)
- Use the embedded FIPS-140-1 validated Microsoft Enhanced Cryptographic Provider (RSAENH.dll) certificate number 238. See http://csrc.nist.gov/cryptval/140-1/1401val2002.htm Certificate #238 for the latest version.
- If a Hardware Random Number Generator is to be used, ensure it is the Intel Hardware Random Number Generator (FWH80802).
- Wait four minutes after initializing the system before making the first service request to ensure the quality of seed material for the DRNG.
- Use unique key container names when using the system. Do not use the default key container to store private keys.
- Configure the TrustConnector Cryptographic Service Provider to not use the TPM option by setting the `UseTPM` flag to 0 in the file cspsetup.ini in the installation directory prior to installing TrustConnector 2.
- Set the `ForceFIPS` flag to 1 in the file cspsetup.ini in the installation directory prior to installing TrustConnector 2.

- Do not use keys generated under a previous version of TrustConnector 2. All keys used in FIPS mode must be generated in FIPS mode or imported into the Module while running in FIPS mode and only have been used while the Module is in FIPS mode.

- Verify that TrustConnector 2 configuration is done according to above enumerated rules to operate the module in FIPS approved mode.

Non-FIPS mode of operation

In non-FIPS mode, the cryptographic module provides non-FIPS approved algorithms as follows:

- RC2
- RC4
- MD5

The following Microsoft CryptoAPI functions are not FIPS approved:

- CryptDeriveKey
- CryptHashSessionKey
- The use of CryptEncrypt with RSA public keys
- The use of CryptDecrypt with RSA private keys

Use of any of the following will put the module in non-FIPS mode:

- Use of RC2, RC4, and MD5 algorithms
- Calling CryptDeriveKey or CryptHashSessionKey functions
- Encrypting or decrypting data with RSA keys
- Requesting a service before four minutes have elapsed after initializing the module
- Running on an operating system other than Microsoft Windows XP Professional Service Pack 2 in single user mode
- Using a Cryptographic Service Provider other than Microsoft Enhanced Cryptographic Provider (RSAENH).
- Configuring the TrustConnector Cryptographic Service Provider to use the TPM option
- Use of a TPM module.

4. Ports and Interfaces

The TrustConnector 2 logical interface is the Microsoft CryptoAPI Application Program Interface (API). Services provided by the module are accessed by function calls. Control Input and Data Input are provided by the variables passed with the function call. These variables are passed on the program stack either directly on the stack or as a pointer on the stack that points to memory allocated in a heap. Both stack and heap are located in RAM. Data Output is provided by variables returned from a function call. As with Control Input and Data Input, these variables are located either on the program stack or in a heap. The Status Output is provided in the return and error codes provided by a function.

All data output is prohibited whenever an error state occurs or during the self-test process.

TrustConnector 2 does not support a maintenance access interface.

5. Identification and Authentication Policy

This section describes the identification and authentication policy of the module.

Assumption of roles

TrustConnector 2 supports a User role and Crypto Officer role. The User role provides the basic services to process data (encryption, decryption and integrity checking), whereas the Crypto Officer role provides the services necessary for testing the module, zeroization of the module and initialization of the module.

TrustConnector 2 does not support a Maintenance role.

The role of the operator of TrustConnector 2 is identified implicitly on the library function being called, as shown in Table 2 in the next section. There is no operator authentication for the assumption of a User role or Crypto Officer role.

Optional Authenticated Access to RSA Private Keys

When an RSA Private Key is either generated by the module or imported, the operator has the option of associating a password with the RSA Private Key. This password is input into the module via a User Interface. The password is initially used to obfuscate the container key that in turn encrypts the RSA Private Key. When the operator uses the RSA Private Key to either sign data or to unwrap a key, the proper password must be provided. If the proper password is not provided, container key will not be properly deobfuscated and thus the RSA Private Key will not be accessible. Note that the process of obfuscation with the password does not provide cryptographic protection or confidentiality of the container key; rather it is used to facilitate operator authentication.

6. Access Control Policy

This section describes the access control policy of the module.

Roles and Services

The services available to each role are described in the following table.

Table 2 – Services Authorized for Roles

Role	Authorized Services
User Role: This role shall provide all of the services necessary to: • Examine and set the attributes of the TrustConnector CSP. • Support data encryption and decryption operations. • Compute hashes and create and verify digital signatures.	• Get CSP Parameters: This service allows the examination of the data that governs the operation of the TrustConnector CSP. • Set CSP Parameters: This service allows the setting of the data that governs the operation of the TrustConnector CSP. • Load Key: This service causes a key that is stored persistently to be loaded into the TrustConnector CSP. • Destroy Key: This service releases a previously acquired handle to the key and zeroizes and frees the memory the key occupied. • Delete Key: This service deletes a stored key container. • Generate Key: This service generates a random key or an RSA key pair. • Generate Random: This service generates random bytes. • Get Key Parameters: This service allows the examination of the data that govern the operations of a key. • Set Key Parameters: This service allows the setting of the data that govern the operations of a key. • Get User Key: This service acquires a handle to one key of a public/private key pair. • Duplicate Key: This service makes an exact copy of a key. • Encrypt Data: This service encrypts plaintext data into ciphertext.

Role	Authorized Services
	• Decrypt Data: This service decrypts ciphertext data. • Create Hash: This service initiates the hashing of a stream of data. It creates a hash object. • Destroy Hash: This service destroys the hash object created by the Create Hash service. • Get Hash Parameters: This service allows the examination of the data that govern the operations of a hash object. • Set Hash Parameters: This service allows the setting of the data that govern the operations of a hash object. • Hash Data: This service hashes a stream of data. • Sign Hash: This service signs data. The data must first be hashed and the hash is signed. • Verify Signature: This service verifies the signature on a hash. • Duplicate Hash: This service is used to make an exact copy of a hash. • Show Status: This service provides the current status of the cryptographic module. • Load Device Key: This service loads the Device Key into the TrustConnector 2 Module from persistent storage. • Import Key: This service imports a key into the TrustConnector CSP using a Microsoft CryptoAPI keyblob. • Export Key: This service exports a key from the TrustConnector CSP using a Microsoft CryptoAPI keyblob.

Role	Authorized Services
Crypto Officer Role: This role shall provide the services necessary for testing the module, zeroization of the module and initialization of the module	• Generate Device Key: This service generates the initial Device Key and will output to persistent storage. • Perform Self-Tests: This service executes the suite of self-tests required by FIPS-140-2. • Zeroize: This service actively destroys all plaintext critical security parameters.

The Show Status service of TrustConnector 2 is indicated by the success or failure of any function call. If the module is in an error state, all calls will fail and the extended error information will be set with the FIPS Error code.

The Perform Self-Tests service is automatically run either when the TrustConnector StrongClient Software Crypto Engine is loaded or if the TrustedCore StrongROM Firmware Crypto Engine is present, when the PC is powered on. The operator can cause this service to be run by rebooting the machine.

The Zeroize service is invoked by calling CryptReleaseContext function on all outstanding contexts and powering down the module.

The following table specifies the inputs and output for each service.

Table 3 - Specification of Service Inputs & Outputs

Service	Control Input	Data Input	Data Output	Status Output
Get CSP Parameters	CryptGetProvParam	Provider handle	Provider parameters	Succeed / Fail
Set CSP Parameters	CryptSetProvParam	Provider handle and provider parameters	None	Succeed / Fail
Load Key	CryptSignHash CryptVerifySignature	Hash to sign Signature to verify	Signature None	Succeed / Fail Succeed / Fail
Destroy Key	CryptDestroyKey	Key handle	None	Succeed / Fail

Service	**Control Input**	**Data Input**	**Data Output**	**Status Output**
Delete Key	CryptAcquireContext	Key handle, Flags	None	Success / Fail
Generate Key	CryptGenKey	Algorithm Id	Key handle	Succeed / Fail
Generate Random	CryptGenRandom	None	Random data	Succeed / Fail
Get Key Parameters	CryptGetKeyParam	Key handle	Key parameters	Succeed / Fail
Set Key Parameters	CryptSetKeyParam	Key handle and key parameters	None	Succeed / Fail
Get User Key	CryptGetUserKey	None	Key handle	Succeed / Fail
Duplicate Key	CryptDuplicateKey	Key handle	Key handle	Succeed / Fail
Encrypt Data	CryptEncrypt	Key handle , Plaintext data	Ciphertext data	Succeed / Fail
Decrypt Data	CryptDecrypt	Key handle , Ciphertext data	Plaintext data	Succeed / Fail
Create Hash	CryptCreateHash	Algorithm Id	Hash object handle	Succeed / Fail
Destroy Hash	CryptDestroyHash	Hash object handle	None	Succeed / Fail
Get Hash Parameters	CryptGetHashParam	Hash object handle	Hash parameters	Succeed / Fail
Set Hash Parameters	CryptSetHashParam	Hash object handle and hash parameters	None	Succeed / Fail
Hash Data	CryptHashData	Hash object handle and data	None	Succeed / Fail

Service	**Control Input**	**Data Input**	**Data Output**	**Status Output**
Sign Hash	CryptSignHash	Hash to be signed	Signature data	Succeed / Fail
Verify Signature	CryptVerifySignature	Hash object handle and signature data	None	Succeed / Fail
Duplicate Hash	CryptDuplicateHash	Hash object handle	Hash object handle	Succeed / Fail
Load Device Key	CryptGenKey CryptSignHash CryptImportKey	Algorithm Id Hash to sign Key data, Decryption Key for Key data	Key handle Signature Key handle	Succeed / Fail Succeed / Fail Succeed / Fail
Import Key	CryptImportKey	Key data, Decryption Key for Key data	Key handle	Succeed / Fail
Export Key	CryptExportKey	Key handle, Encryption Key for Key data	Key data	Succeed / Fail
Generate Device Key	None	None	None	Succeed / Fail
Perform Self-Tests	None	None	None	Succeed / Fail
Show Status	Any function call	None	None	Succeed / Fail
Zeroize	CryptReleaseContext	Provider Name	None	Succeed / Fail

Definition of Critical Security Parameters (CSPs)

The following are the critical security parameters contained in the module:

- Device Key (DK): This key is generated by the TrustConnector 2 Module and is an AES key used to protect the Container Key.

- Container Key: This is an AES key used to protect containers. It is randomly created each time a container is protected and is cryptographically associated with the container. It is also used as an HMAC-SHA-1 key to compute a MAC to ensure the integrity of the container.

- RSA Private Keys: These keys are either imported to TrustConnector 2 by the operator, generated by the operator using TrustConnector 2 functions, or loaded by the operator using TrustConnector 2 functions if the key already exists in a protected container.

- Triple-DES Data Keys: These keys are either imported into TrustConnector 2 by the operator or generated by the operator using TrustConnector 2 functions.

- DES Data Keys: These keys are either imported into TrustConnector 2 by the operator or generated by the operator using TrustConnector 2 functions.

- Customer Password: A password used to access protected RSA Private Keys. It is entered via a user interface for use by TrustConnector 2 and binds the protected RSA Private Key container to the user that possesses this password.

Definition of Public Keys

The following public key is contained in the TrustConnector 2 Module:

- Software Signing Public Key: Used to sign Intermediate Software Signing keys, and to verify signatures on TrustConnector StrongClient Software Crypto Engine program files and configuration data files.

- Intermediate Software Signing Public Key: Used to verify signatures on TrustConnector CSP program files and configuration data files.

- RSA Public Keys: These keys are either imported into TrustConnector 2 by the operator or generated by the operator using TrustConnector 2 functions.

Definition of CSPs Modes of Access

Table 4 defines the relationship between access to CSPs and the different module services. The modes of access shown in the table are defined as follows:

- Create: This operation generates a random cryptographic key.
- Destroy: This operation zeroizes memory for the CSP and frees the memory.
- Delete: This operation causes a stored CSP to be deleted.

- Read: This operation accesses the key to obtain information about the CSP or to use the CSP in an operation.
- Write: This operation writes information into a CSP.
- Import: This operation imports a key or key pair into the module via the logical API.
- Export: This operation exports a key or key pair out of the module via the logical API.
- Load: This operation causes a CSP to be loaded into the module from persistent storage or via a user interface.
- Store: This operation outputs the CSP for persistent storage.

The following table describes how the services performed by each role access the CSP. An "X" means that the service is allowed in that mode.

Table 4 – CSP Access Rights within Roles & Services

Role		Service	Cryptographic Keys and CSPs Access Operation
C.O.	**User**		
	X	Get CSP Parameters	None
	X	Set CSP Parameters	None
	X	Load Key	Load RSA Public Key or RSA Private Key
	X	Destroy Key	Destroy an in memory Triple-DES Data Key, DES Data Key, RSA Public Key or RSA Private Key.
	X	Delete Key	Delete RSA Public Key. Delete RSA Private Key.

Role		Service	Cryptographic Keys and CSPs Access Operation
C.O.	**User**		
	X	Generate Key	Create Triple-DES Data Key, DES Data Key, or RSA Public/Private Key pair. If an RSA Private Key is created: Load Device Key Load Customer Password Create Container Key Read Device Key Read Container Key Read Customer Password Store RSA Private Key and Container Key Destroy RSA Private Key, Container Key, Customer Password, and Device Key
	X	Generate Random	None
	X	Get Key Parameters	Read the parameters associated with a Triple-DES Data Key, or DES Data Key, or RSA Public/Private key pair.
	X	Set Key Parameters	Writes the parameters associated with a Triple-DES Data Key, or DES Data Key, or RSA Public/Private key pair.
	X	Get User Key	None
	X	Duplicate Key	None
	X	Encrypt Data	Read Triple-DES Data Key or DES Data Key.
	X	Decrypt Data	Read Triple-DES Data Key or DES Data Key.
	X	Create Hash	None
	X	Destroy Hash	None
	X	Get Hash Parameters	None
	X	Set Hash Parameters	None
	X	Hash Data	None

Role		Service	Cryptographic Keys and CSPs Access Operation
C.O.	**User**		
	X	Sign Hash	Load Device Key Load Customer Password Load RSA Private Key Load Container Key Read Device Key Read Customer Password Read Container Key Read RSA Private Key Destroy RSA Private Key, Container Key, Customer Password, and Device Key
	X	Verify Signature	Load RSA Public Key Read RSA Public Key Destroy RSA Public Key
	X	Duplicate Hash	None
	X	Load Device Key	Load Device Key
	X	Show Status	None
X		Import Key	Import a Triple-DES Data Key, DES Data Key, RSA Public Key, or RSA Private Key. If an RSA Private Key is imported: Load Device Key Load Customer Password Create Container Key Read Device Key Read Customer Password Read Container Key Store RSA Private Key and Container Key Destroy RSA Private Key, Container Key, Customer Password and Device Key
X		Export Key	Export Triple-DES Data Key, DES Data Key, or RSA Public Key.

Role		Service	Cryptographic Keys and CSPs Access Operation
C.O.	**User**		
X		Generate Device Key	Create Device Key Store Device Key
X		Perform Self-Tests	None
X		Zeroize	Destroy Device Key

7. Operational Environment

The operational environment for each of the TrustConnector 2 configurations, with an Intel Hardware Random Number Generator or without an Intel Hardware Random Number Generator, is a “modifiable operational environment.”

The FIPS-140-2 Area 6 Operational Environment requirements are satisfied in the following ways:

When TrustConnector 2 is operated in FIPS approved mode, the environment is restricted to a single operator mode of operation (i.e., concurrent operators are explicitly excluded).

The module prevents access by other processes to plaintext private and secret keys, CSPs, and intermediate key generation values during the time the cryptographic module is executing/operational using address space separation mechanisms of the operating system. TrustConnector 2 Module relies on the operating system to prevent non-cryptographic processes from stopping the module or starving the module of necessary resources during execution.

TrustConnector 2 software is installed in a manner that protects the software and executable code from unauthorized disclosure and modification. TrustConnector 2 software components are distributed in binary form only, which does not allow decompilation or access to un-compiled source code.

Integrity tests are performed using Power-Up Self Tests Software Integrity Test (see Section 8. Security Rules).

8. Security Rules

The TrustConnector 2 design corresponds to the TrustConnector 2 security rules. This section documents the security rules enforced by the cryptographic module to implement the security requirements of this FIPS-140-2 Level 1 module.

1. TrustConnector 2 is supported on Windows XP Pro Service Pack 2.
2. TrustConnector 2 protects RSA Private Keys in an encrypted form.
3. The module performs the following list of tests. Table 5 lists the tests performed by each TrustConnector 2 component.

   A. Power up Self-Tests: These tests are performed without any operator intervention.

      1. Cryptographic algorithm tests:
         a. AES 128 CBC Known Answer Test
         b. DRNG Known Answer Test
         c. SHA-1 Known Answer Test
         d. DES ECB Known Answer Test
         e. DES CBC Known Answer Test
         f. TDES ECB Known Answer Test
         g. TDES CBC Known Answer Test
         h. TDES 112 ECB Known Answer Test
         i. TDES 112 CBC Known Answer Test
         j. HMAC-SHA-1 Known Answer Test
         k. RSA sign/verify power up test
         l. RSA wrap/unwrap key power up test
      2. Software Integrity Tests
         a. Software integrity test via DESMAC checksum of the Microsoft RSAENH DLL image
         b. Software integrity test via HMAC-SHA-1 error detection code of TrustConnector CSP, TrustConnector StrongClient Software Crypto Engine and TrustedCore StrongROM Firmware Crypto Engine images.

   B. Conditional Self-Tests: These tests are performed during the appropriate services.

      1. Continuous Random Number Generator (RNG) test – initiated at random number generation and performed by all random number generators (approved and non-approved).
      2. RSA pair-wise consistency test – initiated at RSA key pair generation

Table 5 – Component Self-Tests

Self-Test Performed	TrustedCore StrongROM Firmware Crypto Engine	TrustConnector StrongClient Software Crypto Engine	**TrustConnector Cryptographic Service Provider**	**Microsoft Enhanced Cryptographic Service Provider**
AES 128 CBC KAT	X	X		X
DRNG KAT	X	X		
SHA-1 KAT	X	X		X
DES ECB KAT				X
DES CBC KAT				X
TDES ECB KAT				X
TDES CBC KAT				X
TDES 112 ECB KAT				X
TDES 112 CBC KAT				X
HMAC-SHA-1 KAT[1]				X
RSA sign/verify power up test			X	X
RSA wrap/unwrap key power up test			X	
HMAC-SHA-1 software integrity test	X	X	X	
DESMAC software integrity test				X
DRNG continuous test	X	X		X
RSA pair-wise consistency test			X	X

[1] The TrustConnector CSP, TrustConnector StrongClient Software Crypto Engine, and TrustedCore StrongROM Firmware Crypto Engine use HMAC-SHA-1 for the software integrity test and thus do not perform a separate HMAC-SHA-1 known answer test.

9. Physical Security Policy

TrustConnector 2 is intended to operate on a general purpose computer, which is defined as a Multi-Chip Standalone device. The general purpose computer shall be comprised of production grade components and a production grade enclosure.

10. Mitigation of Other Attacks Policy

The module is not designed to mitigate any other attacks.

11. References

This section contains informative references that provide helpful background information.

[FIPS-140-2] "Security Requirements for Cryptographic Modules" Version 2, May 25, 2001.
http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[FIPS-180-2] "Secure Hash Standard" Version 2, August 1, 2002.
http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf

[FIPS-186-2] "Digital Signature Standard (DSS)" Version 2, January 27, 2000.
http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf

[FIPS-197] "Advanced Encryption Standard (AES)" November 26, 2001.
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[FIPS-198] "The Keyed-Hash Message Authentication Code (HMAC)" March 6, 2002. File updated April 8, 2002.
http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf

[Microsoft CryptoAPI] See:
http://msdn.microsoft.com/library/en-us/security/security/cryptography_reference.asp

[RSAENH] Enhanced Cryptographic Provider (RSAENH)
http://csrc.nist.gov/cryptval/140-1/140sp/140sp238.pdf

12. Definitions and Acronyms

The following paragraphs define the acronyms used in this document.

AES	Advanced Encryption Standard secret key algorithm – See [FIPS-197]
ANSI	American National Standards Institute
API	Application Program Interface
CBC	Cipher Block Chaining mode
CSP	Cryptographic Service Provider or Critical Security Parameters
DES	Data Encryption Standard – See [FIPS-46-3]
DRNG	Deterministic Random Number Generator
DK	Device Key
ECB	Electronic Codebook mode
EMI	Electromagnetic Interference
EMC	Electromagnetic Compatibility
FIPS	Federal Information Processing Standards of NIST
HMAC-SHA-1	Hash-based Message Authentication Code based on SHA-1 – See [FIPS-198]
KEK	Key Encryption Key
MD5	Message Digest algorithm 5 by Professor Ronald Rivest
NIST	National Institute of Standards and Technologies
PKI	Public Key Infrastructure
RC2	Rivest Cipher algorithm 2 by Professor Ronald Rivest
RC4	Rivest Cipher algorithm 4 by Professor Ronald Rivest
RSA	Rivest Shamir Adleman public key algorithm
SHA-1	Secure Hash Algorithm revision 1 – See [FIPS-180-2]
TDES	Triple DES – See [FIPS-43-3]
TPM	Trust Platform Module
X9	ISO financial standards group
X9.31	X9 standard regarding public key algorithms