bTrade, LLC

bTrade Security Module, V1.0

FIPS 140-2 Non-Proprietary Security Policy

Level 1 Validation

April 2011

© 2011 bTrade, LLC. All rights reserved. www.btrade.com

This document may be freely reproduced and distributed whole and intact including this copyright notice.

Table Of Contents

1 INTRODUCTION

1.1 PURPOSE

This document is the nonproprietary security policy for the **bTrade Security Module v1.0**. This

document was prepared as part of the Federal Information Processing Standard (FIPS) 140-2 Level 1 validation process.

FIPS 140-2, *Security Requirements for Cryptographic Modules*, describes the requirements for cryptographic modules. For more information about the FIPS 140-2 standard and the cryptographic module validation process see http://csrc.nist.gov/groups/STM/cmvp/index.html

1.2 REFERENCES

This Security Policy describes how this module complies with the eleven sections of the

STANDARD:

For more information on the FIPS 140-2 standard and validation program please refer to the NIST website at http://csrc.nist.gov/groups/STM/cmvp/index.html

1.3 DOCUMENT HISTORY

Authors	Date	Version	Comment
Hatem El-Sebaaly Ryan Saldana	June 1, 2009	1.0	Initial Document.
Hatem El-Sebaaly Ryan Saldanha	July 16, 2009	1.1	Incorporated new changes.
Hatem El-Sebaaly Ryan Saldanha	July 27, 2009	1.2	Incorporated new changes.
Hatem El-Sebaaly	Jan 29, 2010	1.3	Incorporated CMVP corrections.

Clifton Gonsalves	Dec 08, 2010	1.4	Added iSeries (AS400) and zSeries(Mainframe).
Clifton Gonsalves	Apr 13, 2011	1.5	Corrections in nomenclature for iSeries(AS400) and zSeries(Mainframe).

2 MODULE SPECIFICATION

The **bTrade Security Module v1.0** (hereafter referred to as the Module) is a software library supporting FIPS approved cryptographic algorithms. For the purposes of the FIPS 140-2 level 1 validation, the **bTrade Security Module v1.0** is a single module file named *libtdfips.so on* Linux, Solaris and AIX, *libtdfips.sl on HP-UX* and *tdfips.dll on* Microsoft Windows. This module provides a C application program interface (API) for use by other processes that require cryptographic functionality.

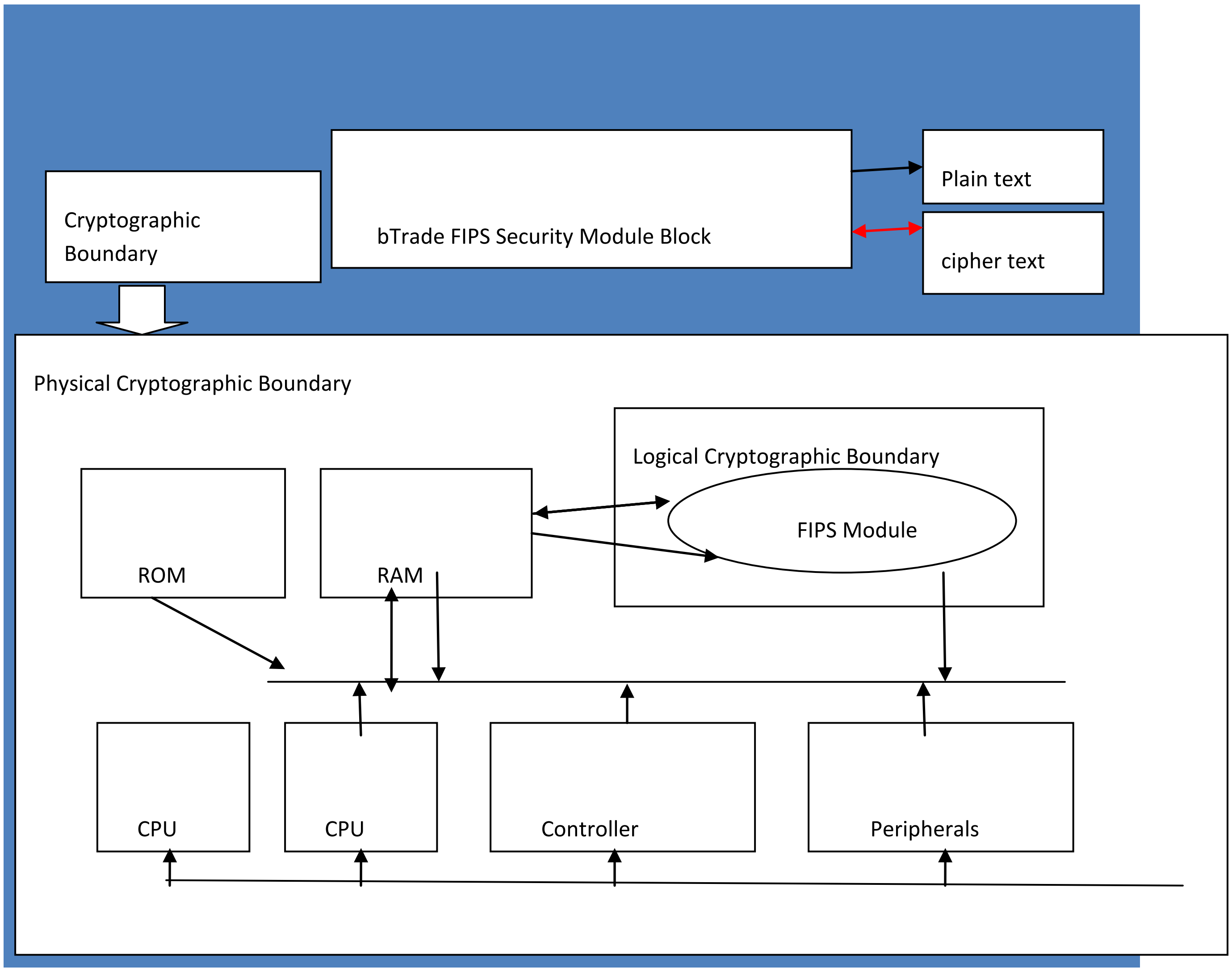
For FIPS 140-2 purposes the Module is classified as a multichip standalone module and is a shared object. The *logical* cryptographic boundary of the Module is the *btradesecure* module. The *physical* cryptographic boundary of the Module is the enclosure of the computer system on which it is executing. The Module performs no communications other than with the process that calls it. It makes no network or inter-process connections and creates no files.

The Module was tested on the following platforms:

1. IBM AIX 6.1 on IBM Power4 Processor (module is named *libtdfips.so*)
2. HP-UX 11.3 on HP PA-7300 RISC Processor (module is named *libtdfips.sl*)
3. SUN Solaris 10 on SUN UltraSPARCIIIi Processor (module is named *libtdfips.so*)
4. Microsoft Windows Vista on Intel Core2 Quad Processor (module is named *libtdfips.dll*)
5. IBM System z9 with IBM z/OS 1.10 (module is named TDFIPS)
6. IBM POWER6 with IBM i 6.1 (module is named TDFIPS)

The module implements Triple-DES, AES, SHS, HMAC, RNG, RSA and DSA (signing/verification) algorithms in the approved mode. The module also implements MD5 in the non approved mode. Diffie-Hellman is allowed for use as well.

The module is distributed as a shared object.


Cryptographic Boundary
bTrade FIPS Security Module Block
Plain text
cipher text
Physical Cryptographic Boundary
Logical Cryptographic Boundary
FIPS Module
ROM
RAM
CPU
CPU
Controller
Peripherals

2.1 ROLES AND SERVICES

The Module meets all FIPS 140-2 level 1 requirements for Roles and Services, implementing both Crypto-User and Crypto-Officer roles. As allowed by FIPS 140-2, the Module does not support user authentication for those roles. Only one role may be active at a time and the Module does not allow concurrent operators.

The User and Crypto-Officer roles are implicitly assumed by the entity accessing services implemented by the Module. The Crypto-Officer can install and initialize the Module. The Crypto-Officer role is implicitly entered when installing the Module or performing system administration functions on the host operating system.

User Role: Executing the Module and calling any of the API functions. This role has access to all of the services provided by the Module.

Crypto Officer Role: Installation of the Module on the host computer system. This role is assumed implicitly when the system administrator installs the bTrade application package containing the application(s) and module library file.

TABLE.1 SERVICES

Service	Role	CSP	Access
Module Installation	**Crypto Officer**	**None**	**read/write/execute**
Symmetric Encryption/Decryption	**User, Crypto Officer**	**Symmetric Key** **AES, TDES**	**read/write/execute**
Key Establishment	**User, Crypto Officer**	**Asymmetric private Key RSA** **Diffie-Hellman Shared Secret**	**read/write/execute**
Digital signature	**User, Crypto Officer**	**Asymmetric private Key RSA, DSA**	**read/write/execute**
Asymmetric key generation	**User, Crypto Officer**	**Asymmetric private Key RSA, DSA**	**read/write/execute**

Keyed Hash (HMAC)	**User, Crypto Officer**	**HMAC SHA-1 key** **HMAC SHA-1**	**read/write/execute**
Message digest (SHS)	**User, Crypto Officer**	**None** **SHA-1, SHA-2**	**read/write/execute**
Random number generation	**User, Crypto Officer**	**Seed key** **Seed** **AES**	**read/write/execute**
Show Status	**User, Crypto Officer**	**None**	**Execute**
Module initialization	**User, Crypto Officer**	**None**	**Execute**
Self tests	**User, Crypto Officer**	**None**	**Execute**
Zeroize	**User, Crypto Officer**	**Symmetric key asymmetric key** **Seed key**	**Execute**

2.2 PORTS AND INTERFACES

The physical ports of the Module are the same as the computer system on which it is executing. The logical interface is a C application program interface (API).

The Data Input interface consists of the input parameters of the API functions. The Data Output interface consists of the output parameters of the API functions. The Control Input interface consists of the actual API functions. The Status Output interface includes the return values of the API functions.

TABLE.2 PORTS AND INTERFACES

FIPS Interface	Physical Port	Module Interface
Data Input	**Ethernet ports**	**API input parameters**
Data Output	**Ethernet ports**	**API output parameters**
Control Input	**Keyboard, Serial port, Ethernet port**	**API function calls**
Status Output	**Keyboard, Serial port, Ethernet port**	**API return codes**
Power Input	**PCI Compact Power Connector**	**N/A**

2.3 SELF TESTS

The Module performs both power up self tests at module initialization and conditional tests during operation. Input, output, and cryptographic functions cannot be performed while the Module is in a self-test or error state as the module is single threaded and will not return to the calling application until the power up self tests are complete. If the power-up self-tests fail subsequent calls to the module will fail and thus no further cryptographic operations are possible.

TABLE.3 POWER-UP SELF TESTS

Algorithm	Test
AES	KAT
Triple-DES	KAT
DSA	Pair wise consistency test, sign/verify
RSA	KAT
PRNG	KAT
HMAC-SHA-1	KAT
HMAC-SHA-224	KAT
HMAC-SHA-256	KAT
HMAC-SHA-384	KAT
HMAC-SHA-512	KAT
SHA-1	Performed as part of HMAC-SHA-1 KAT
SHA-224	Performed as part of HMAC-SHA-224 KAT
SHA-256	Performed as part of HMAC-SHA-256 KAT
SHA-384	Performed as part of HMAC-SHA-384 KAT
SHA-512	Performed as part of HMAC-SHA-512 KAT
Module integrity	HMAC-SHA-1 software integrity test

TABLE.4 CONDITIONAL SELF TESTS

Algorithm	Test
DSA	**Pair wise consistency**
RSA	**Pair wise consistency**
PRNG	**Continuous test**

A single initialization call, FIPS_mode_set(), is required to initialize the Module for operation in the FIPS 140-2 Approved mode. When the Module is in FIPS mode all security functions and cryptographic algorithms are performed in Approved mode.

The FIPS mode initialization is performed when the application invokes the FIPS_mode_set() call which returns a “1” for success and “0” for failure. Interpretation of this return code is the responsibility of the host application. Prior to this invocation the Module is uninitialized in the non FIPS mode by default. The FIPS_mode_set() function verifies the integrity of the runtime executable using a HMAC-SHA-1 digest computed at build time. If this computed HMAC-SHA-1 digest matches the stored known digest then the power-up self-test, consisting of the algorithm specific Pair wise Consistency and Known Answer tests, is performed. If any component of the power up self test fails an internal global error flag is set to prevent subsequent invocation of any cryptographic function calls. Any such power up self test failure is a hard error that can only be recovered by reinstalling the Module. If all components of the power up self test are successful then the Module is in FIPS mode. The power up self tests may be performed at any time with a separate function call, FIPS_selftest(). This function call also returns a “1” for success and “0” for failure, and interpretation of this return code is the responsibility of the host application.

A power up self test failure can only be cleared by a successful FIPS_mode_set() invocation. No operator intervention is required during the running of the self tests.

2.4 MITIGATION OF OTHER ATTACKS

The Module does not contain additional security mechanisms beyond the requirements for FIPS 1402 level 1 cryptographic modules.

2.5 PHYSICAL SECURITY

The Module is comprised of software only and thus does not claim any physical security.

3. CRYPTOGRAPHIC KEY MANAGEMENT

3.1 KEY GENERATION

The Module supports generation of DSA, and RSA public/private key pairs. The Module employs an ANSI X9.31 compliant random number generator for creation of asymmetric and symmetric keys.

3.2 KEY STORAGE

Public and private keys are provided to the Module by the calling process, and are destroyed when released by the appropriate API function calls. The Module does not perform persistent storage of keys.

3.3 KEY ACCESS

An authorized application as User has access to all key data generated during the operation of the Module.

3.4 KEY PROTECTION AND ZEROIZATION

Keys residing in internally allocated data structures can only be accessed using the Module defined API. The operating system protects memory and process space from unauthorized access. Zeroization of sensitive data is performed automatically by API function calls for intermediate data items, and on demand by the calling process using Module provided API function calls provided for that purpose.

Only the process that creates or imports keys can use or export them. No persistent storage of key data is performed by the Module. All API functions are executed by the invoking process in a non overlapping sequence such that no two API functions will execute concurrently.

The calling process can perform key zeroization of keys by calling an API function.

3.5 CRYPTOGRAPHIC ALGORITHMS

The Module supports the following FIPS approved or allowed algorithms:

TABLE.6 FIPS APPROVED OR ALLOWED ALGORITHMS

Algorithm	Validation Certificate	Usage	Keys/CSPs
AES	#1064	encrypt/decrypt	AES keys 128, 192, 256 bits
Triple-DES	#796	encrypt/decrypt	Triple-DES keys, 168 bits
DSA	#352	sign ad verify	DSA keys 1024 bits
PRNG (ANSI X9.31 Appendix A.2.4 using AES)	#601	random number generation	PRNG seed value is 128 bits; seed key values are 128 bits; 192 bits and 256 bits

RSA (X9.31,PKCS #1.5, PSS)	#504	sign and verify	RSA keys 1024 to 16384 bits
RSA for Key Wrapping	N/A	key wrapping	RSA (key wrapping; key establishment methodology provides 80 to 256 bits of encryption strength)
SHA-1	#1007	hashing	N/A
SHA-224	#1007	hashing	N/A
SHA-256	#1007	hashing	N/A
SHA-384	#1007	hashing	N/A
SHA-512	#1007	Hashing	N/A
HMAC-SHA-1	#599	message integrity	HMAC key
HMAC-SHA-224	#599	message integrity	HMAC key
HMAC-SHA-256	#599	message integrity	HMAC key
HMAC-SHA-384	#599	message integrity	HMAC key
HMAC-SHA-512	#599	message integrity	HMAC key
Diffie-Hellman	N/A	Key Agreement	The Module only provides functions that implement Diffie-Hellman primitives. The shared secret provides between 80 and 219 bits of encryption strength.

The Module supports the following non FIPS approved algorithms:

TABLE.7 NON-FIPS APPROVED ALGORITHMS

Algorithm	Usage	Keys/CSPs
MD5	Hashing	N/A

4. DESIGN ASSURANCE

The Module is managed in accordance with the established configuration management and source version control procedures of bTrade, LLC.

4.1 SOURCE CODE CONTROL

The source code revisions are maintained in a Perforce repository with public read access but with write access restricted to the engineering team. Individually packaged revisions are released periodically in installable form. The integrity of the Module is based on HMAC-SHA1.

4.2 *APPLICATION MANAGEMENT OF CRITICAL SECURITY PARAMETERS*

IDENTIFYING CSPS

All CSPs must be created, stored, and destroyed in an approved manner as described by Reference 1, FIPS 140-2. CSPs are those items of information which must be protected from disclosure, such as symmetric keys, asymmetric private keys, etc. Note that the application designer and end user/system administrator/Crypto Officer share a responsibility for protection of CSPs; the former to include appropriate technical protections and the latter to install and configure the application correctly. Technical protections include checks to require that files storing CSPs have appropriate permissions (not group writable or world readable, for example). Administrative protections include installation of the runtime software (executables and configuration files) in protected locations. End users have a responsibility to refrain from comprising CSPs (as by sending a password in clear text or copying an encryption key to an unprotected location).

KEY GENERATION

Secret keys which are established must be wrapped with RSA by the calling application (the application using the Module to perform cryptographic operations) before being output. For key wrapping using RSA, the key used to wrap the transported key must be larger than the key that is being transported. The minimum key size that must be used for this operation is 1024 bits which provides 80 bits of encryption strength.

STORAGE OF CSPS

The Module does not store any critical security parameters (CSPs) in persistent media; while the Module is initialized any CSPs reside temporarily in RAM and are destroyed at the end of the session. Any keys or other CSPs otherwise stored in persistent media must be protected in accordance with FIPS requirements in Reference 1, FIPS 140-2.

DESTRUCTION OF CSPS

When no longer needed, CSPs contained within the application must be deleted by overwriting in a way that will make them unrecoverable. The fips_rand_bytes() function in the Module can be used to generate random data to overwrite the storage location of a CSP.

Note the OPENSSL_cleanse() function call which is typically used to perform key wiping functions is not part of the Module. This function overwrites a memory storage area to ensure destruction of data, using the random data generation functions of the Module.

5. CRYPTO OFFICER GUIDANCE

The crypto officer is responsible for installing the FIPS module. Installation instructions which are more detailed than this Security Policy are provided in the bTrade TDNgine User Guide and bTrade TDAccess User Guide.

5.1 SETUP AND INITIALIZATION PROCESS

The FIPS module is shipped as part of various bTrade TD product offerings. It is automatically installed along with the said products. As part of this installation a file named **<fips-module-name>.tdfipshmac** is also installed. This file will contain the sha1 HMAC signature of the FIPS module to guarantee that the module is not compromised. The HMAC key is kept confidential. There is no manual installation requirements for this module. The operator need only install the bTrade product package which contains it.

5.2 SECURE OPERATION

The tested operating systems segregate user processes into separate process spaces. Each process space is an independent virtual memory area that is logically separated from all other processes by the operating system software and hardware. The Module functions entirely within the process space of the process that invokes it, and thus satisfies the FIPS 140-2 requirement for a single user mode of operation.

The Module is installed as part of the bTrade TD suite of products.

Upon initialization[6] of the Module, the module will run its power-up self-tests. Successful completion of the power-up self-tests ensures that the module is operating in the FIPS mode of operation.

As the Module has no way of managing keys, any keys that are input or output from applications utilizing the module must be input or output in encrypted form using FIPS approved algorithms. The self tests can be called on demand by reinitializing the module using the FIPS_mode_set() function call, or alternatively using the FIPS_selftest() function call.