Pragma Systems Crypto Module
FIPS Security Policy

Version 1.0.0.12
August 8, 2014

Revision History

Date	Author	Notes
11/5/2009	D. Kulwin	Initial Draft
12/17/2009	D. Kulwin	Update to reflect code
1/6/2010	D. Kulwin	Updates for TDR_001
1/22/2010	D. Kulwin	Updates for TDR_002: - Section 5 changed per recommendation - Table 4 and 7 made more consistant. - Added Initialization Vectores as a critical paramater
1/28/2010	D. Kulwin	Updates for TDR_003 - Section 10 modified fo TE10.03.01 - Table 4 modified for TE15.05.01 - Section 5 modified for TE14.05.02 - Table 7 modified for TE14.07.0 and TE01.15.01
2/5/2010	D. Kulwin	Updates for TDR_004
2/12/2010	D. Kulwin	Update version to rev 5
2/18/2010	D. Kulwin	Updates for TDR_007 - Removed support for 3DES 2-Key - Update version to 1.0.0.6
2/25/2010	D. Kulwin	Updates for TDR_008 - Added EMI/EMC section - Update version to 1.0.0.7
3/3/2010	D. Kulwin	Updates for TDR_009 - Added Consistency Test section - Update version to 1.0.0.8
3/12/2010	D. Kulwin	Updates for TDR_010 - Explicity state tested Oss - Refine conditional test wording - Update version to 1.0.0.9
3/17/2010	D. Kulwin	Updates for TDR_011 - Correct typo in section 7 - File protection information i added to section 12 - Fix table 7 for DSA Key Generation - Added info about setting FIPS mode in section 3.1 - Update version to 1.0.0.10
3/31/2010	D. Kulwin	Updates for TDR_012 - Add Service to API table - Update table 7 (now table 8) to reflect new services - Update version to 1.0.0.11
4/10/2010	D. Kulwin	Misc Updates - Expanded “Approved mode of operation to explicitly refer to the 256,384 and 512 variants of the SHA-2 hash family. - Update version to 1.0.0.12
10/20/2010	D. Kulwin	Updates for TDR_013 and TDR_014

		- Change "Approved mode of operation" to clarify that RSA KeyGen is not accessible in FIPS mode. - Updated table in "Approved mode of operation" - Added password length column in "Strengths of Authentication Mechanism" table. - Expanded section "Entering Approved Mode" to include more information on CAPI loading and selftest execution as well as detailing the module API call SetMode() used to change modes. - Changed Figure 1 to include CAPI modules. - Added Ports and Interfaces table to "Ports and Interfaces - Changed wording in Section 7 and added CC statement. - Clarified RSA KeyGen relating to FIPS/NON-FIPS modes in "Identification and Authentication Policy" tables. - Added details to "Operational Environment". - Added section "Module Error Conditions" - Update version to 10.0.0.13
11/5/2010	D. Kulwin	Correct Diffie-Hellman encryption strength
11/5/2010	D. Kulwin	Updates for TDR_015 - Add CMVP MS CAPI Certificate numbers - Update Design Assurance references to Level 1 - Figure changes: CAPI modules included in both operating system and pragma cryptographic module boxes. - Change wording relating to cryptographic boundaries in section 1. - Changed 'evaluated platform' to 'validated platform in section 7. - Correct table references in section 8. - Correct Diffie-Hellman encryption strength wording in section 3.2 - Update version to 10.0.0.14 Update footer copyright to 2010
11/9/2010	D. Kulwin	Updates for TDR_015 ademddem - Change cryptographic boundary wording - Correct wording for Diffie-Hellman encryption strength. - Update version to 10.0.0.15
11/15/2010	D. Kulwin	Updates for TDR_016 - Update version to 10.0.0.16
1/6/2011	D. Kulwin	Updates for TDR_017 - Misc. wording changes - Change level 2 references to level 1 references - Update copyright date. - Update version to 10.0.0.17
1/6/2011	D. Kulwin	Updates for TDR_017

		- Misc. wording changes - Change level 2 references to level 1 references - Update copyright date. - Update version to 10.0.0.17
8/8/2014	D. Kulwin	Adding vendor affirmed platforms: - Microsoft Windows 8.1 - Microsoft Windows Server 2012 R2

Table of Contents

Table of Figures

Table of Tables

1 Introduction

This security policy defines all security rules under which the Pragma Systems Cryptographic Module (Module) must operate and enforce, including rules from relevant standards such as FIPS 140-2. The module complies with all FIPS 140-2 level 1 requirements.

The Module is a *cryptographic software application* that operates as a multi-chip standalone cryptographic module. The physical boundary is the hardware platform, on which the Module is installed. The Pragma Cryptographic Module dynamic link library (DLL) and MS CAPIs RSAENH and DSSENH fall within the cryptographic boundary. See Table 1 for the relevant MS CAPI module certificates. The module is supported on Microsoft Windows 2003 Server, Microsoft Windows 2008, Microsoft Windows 2008 R2, Microsoft Windows Vista, and Microsoft Windows 7. The FIPS 140-2 validation was conducted on the following platforms: Microsoft Windows 2003 Server, Microsoft Windows 2008 Server and Microsoft Windows Vista.

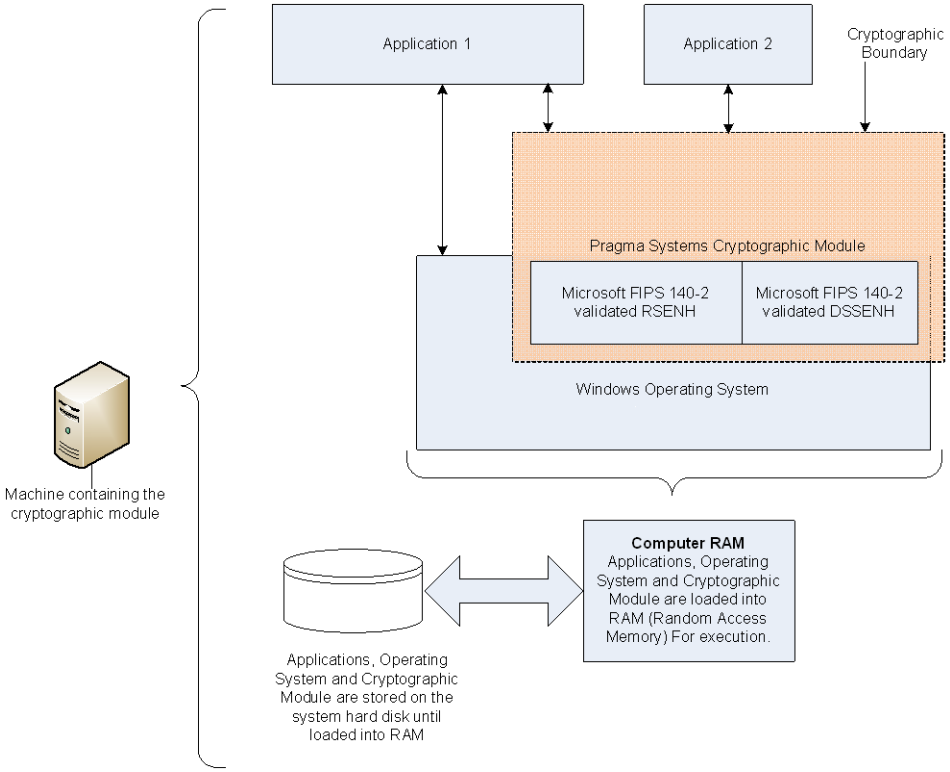

Figure 1 Cryptographic Module Interface Diagram

The Pragma Cryptographic Module relies on Microsoft validated CAPI Modules RSAENH and DSSENH. Below is a list of the relevant certificate numbers and the associated FIPS CMVP table entries:

Certificate Number	Module Title	Microsoft Operating System
1012	**Windows Server 2003 Enhanced Cryptographic Provider (RSAENH)** (Software Version: 5.2.3790.4313)	Windows Server 2003
1010	**Windows Server 2008 Enhanced Cryptographic Provider (RSAENH)** (Software Versions: 6.0.6001.22202 and 6.0.6002.18005)	Windows Server 2008
1002	**Windows Vista Enhanced Cryptographic Provider (RSAENH)** (Software Versions: 6.0.6001.22202 and 6.0.6002.18005)	Windows Vista
1009	**Windows Server 2008 Enhanced DSS and Diffie-Hellman Cryptographic Provider (DSSENH)** (Software Versions: 6.0.6001.18000 and 6.0.6002.18005)	Windows Server 2008
1003	**Windows Vista Enhanced DSS and Diffie-Hellman Cryptographic Provider (DSSENH)** (Software Versions: 6.0.6001.18000 and 6.0.6002.18005)	Windows Vista
875	**Windows Server 2003 Enhanced DSS and Diffie-Hellman Cryptographic Provider (DSSENH)** (Software Version: 5.2.3790.3959)	Windows Server 2003

Table 1 CMVP MS CAPI certificate numbers

2 Security Level

The cryptographic module meets the overall requirements applicable to FIPS 140-2 for the specified level.

SECURITY REQUIREMENTS SECTION	LEVEL
Cryptographic Module Specification	1
Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1

Mitigation of Other Attacks	N/A

Table 2 Module Security Level Specification

3 Modes of Operation

3.1 Approved mode of operation

The module supports a FIPS Approved mode of operation. RSA KeyGen is not accessible in the FIPS Approved mode of operation since it is non-compliant to FIPS 140-2. The user must explicitly set the module into FIPS Approved mode by calling the module API SetMode() with the FIPS_MODE constants set prior to using the module in a FIPS Approved mandated environment. The following FIPS Approved algorithms are supported:

ALGORITHM	CAVP CERTIFICATES
AES (CBC mode, E/D; 128, 192, and 256)	739, 818
Triple-DES (3-key TCBC mode; E/D)	656, 691
HMAC (SHA-1, SHA-256, SHA-384, SHA-512)	407, 408, 452
SHS (SHA-1, SHA-256, SHA-384, SHA-512)	753, 816
RSA (SIG Gen, Sig Verify)	354, 355, 395
DSA (Key Gen, SIG Gen, Sig Verify)	221, 281, 282

Table 3 Algorithm CAVP Certificates

In addition, RNG (Certs. #314, #435 and #470) and DRNG (SP 800-90, vendor affirmed) provided by MS CAPIs are used by the module.

3.2 Non-FIPS Approved Algorithms

Within the FIPS Approved mode of operation, the module supports the following allowed algorithms:

- Diffie-Hellman for SSH v2 (key agreement; key establishment methodology provides between 80 and 150 bits of encryption strength; non-compliant less than 80-bits of encryption strength)

In addition to the above algorithm, the following algorithms are available in the non-FIPS Approved mode of operation:

- RSA KeyGen
- MD5 Hashing

4 Module Error Conditions

The module enters the error state when an error has been encounterd.

The operator of the module can respond to error conditions by trying the API call again, by executing the selftests on demand or by reloading the module.

5 Ports and Interfaces

The physical ports of the module are provided by the general purpose computer on which the module is installed. The logical interfaces are defined as the API or the cryptographic module. The module's API supports the following logical interfaces:

FIPS 140-2 INTERFACE	LOGICAL INTERFACE
Data Input Interface	Input parameters to all functions that accept input from Crypto-Officer or User entities
Data Output Interface	Input parameters from all functions that return values from Crypto-Officer or User entities
Control Input Interface	All API functions that are input into the Module by the Crypto-Officer and User entities
Status Output Interface	Information returned via exceptions (return/exit codes) to Crypto-Officer or User entities.

Table 4 Ports and Interfaces

The logical interfaces do not map to the physical ports, and the only entity accessing the logical interfaces is the application that loaded a specific instance of the Module DLL.

5.1 Entering Approved Mode

The module contains an API (SetMode()) that switches the Module to Approved mode. Calling this API will force the enforcement of only using approved and allowed algorithms. For example, to set the module to FIPS approved mode execute the SetMode() api with the FIPS_MODE enumeration value:

SetMode(FIPS_MODE);

To set non-FIPS mode:

SetMode(NON_FIPS_MODE);

The SetMode() interface call only works once during the module execution. In order to change the mode once it has been initialized, the Module should be unloaded using the Win32 FreeLibrary() call. The module can then be reloaded using the Win32 LoadLibrary() call and the SetMode() call can be invoked to set the moduled to the desired mode.

See the User's Guide for more details on invoking the Module API.

6 Consistency Tests

In addition to power-on and on-demand selftests, a conditional DSA pair-wise consistency test is performed during the creation of DSA keys. If the pair-wise consistency test fails, the key generation call fails.

The module performs a software integrity check at power-up or on demand by verifying a signed (RSA, 1024-bit key) hash (SHA-1) contained in the module's PKI certificate. Modification of any component will cause the module to enter the error state with an integrity failure.

7 EMI / EMC

Although the module consists entirely of software, the FIPS 140-2 validated platform is a standard GPC, which has been tested for and meets applicable Federal Communications Commission (FCC) EMI and EMC requirements for business use as defined in Subpart B of FCC Part 15.

8 Identification and Authentication Policy

The authentication of users is provided by the host operating system. The authentication mechanism is provided by the host Operating System. Proper operation of the module requires that the host Operating System be configured to enforce a password length of at least six characters. The module relies on the Operating System to distinguish between an operator assuming the User role or Crypto Officer role. An operator with Administrator privileges to the Operating System assumes the Crypto Officer role. Table 5 lists these roles along with their required identification and authentication techniques. Table 6 outlines each authentication mechanism and the associated strengths.

ROLE	TYPE OF AUTHENTICATION	AUTHENTICATION DATA
User	Logon	Password
Cryptographic Officer	Logon	Password

Table 5 Roles and Required Identification and Authentication

AUTHENTICATION MECHANISM	STRENGTH OF MECHANISM	MAXIMUM PASSWORD LENGTH
Password	Each password is at least six characters in length. Characters are chosen from a fifty-two character set. The probability of a successful random attempt is less than 1/52^6, which is less than 1/1,000,000. Assuming that no password lockout settings were configured, that no delay is configured between	Windows Server 2003, Vista, Windows Server 2008 - 256 characters Note, that password minimum length and complexity rules are

	password attempts, and that an attacker could attempt 100 password entries per minute, the probability of successfully authenticating to the module within one minute through random attempts is 100/(52^6), which is less than one in 100,000.	configured by the domain or system administrator.

Table 6 Strengths of Authentication Mechanism

The default windows authentication passwords mechanisms will be fine. However, the users MUST NOT modify the operating system in such a way that that the passwords during authentication are not obscured (e.g. no visible display of characters when entering a password). Additionally, the operating MUST NOT be modified in such a way that the feedback provided to the operator during an attempted authentication shall weaken the strength of the authentication mechanism.

All of the services provided by the module are authenticated since the OS provides the authentication and no service is available without logging into the OS.

Table 7 shows the mapping from Authorized Services to module API calls. The sequence of calls is prescribed by the Users manual and each API call is no more granular than the corresponding validated MS CAPI call.

AUTHORIZED SERVICES	ASSOCIATED API CALLS
AES Encryption	• CIPHEREncrypt()
AES Decryption	• CIPHERDecrypt()
AES IV Import	• CIPHERSetIV()
AES Key Import	• CIPHERInit()
DH Key Generation	• DHGenerateKey()
DH Key Exchange	• DHComputeSecret() • DHDeriveSSHValues() • DHGenerateKey() • DHGetCipherKeys() • DHGetHMAC() • DHGetPublicKey() • DHGetSharedSecret()
DSA Key Export	• DSAExportKeys()
DSA Key Generation	• DSAGenKeys()
DSA Key Import	• DSAImportPublicKey() • DSAImportPrivateKey()
DSA Signature Generation	• DSASignData()
DSA Signature Verification	• DSAVerifySig()
HMAC-SHA1 Message Authentication	• HMACFinal() • HMACInit() • HMACSetup()

	• HMACTearDown() • HMACUpdate()
HMAC-SH256 Message Authentication	• HMACFinal() • HMACInit() • HMACSetup() • HMACTearDown() • HMACUpdate()
HMAC-SH384 Message Authentication	• HMACFinal() • HMACInit() • HMACSetup() • HMACTearDown() • HMACUpdate()
HMAC-SH512 Message Authentication	• HMACFinal() • HMACInit() • HMACSetup() • HMACTearDown() • HMACUpdate()
Resource Initialization/Cleanup	• CIPHERFree() • CIPHERInit() • DHFree() • DHInit() • DSAFreeKeys() • HASHFree() • HASHInit() • RSAFreeKeys()
RSA Key Export	• RSAExportKeys()
RSA Key Import	• RSAImportPublicKey() • RSAImportPrivateKey()
RSA Signature Generation	• RSASignData()
RSA Signature Verification	• RSAVerifySig()
Self-tests	• StartSelfCheck()
Set Module Mode	• SetMode()
SHA-1	• HASHFinal() • HASHFree() • HASHInit() • HASHUpdate()
SHA-256	• HASHFinal() • HASHFree() • HASHInit() • HASHUpdate()
SHA-384	• HASHFinal() • HASHFree() • HASHInit() • HASHUpdate()
SHA-512	• HASHFinal()

	• HASHFree() • HASHInit() • HASHUpdate()
Show Status	• GetMode() • GetModuleVersion() • GetState()
TDES Encryption	• CIPHEREncrypt()
TDES Decryption	• CIPHERDecrypt()
TDES IV Import	• CIPHERSetIV()
TDES Key Import	• CIPHERInit()

Table 7 Service to API Call Mapping

9 Access Control Policy

ROLE	AUTHORIZED SERVICES
User	• AES Encryption • AES Decryption • AES IV Import • AES Key Import • DH Key Generation • DH Key Exchange • DSA Key Export • DSA Key Generation • DSA Key Import • DSA Signature Generation • DSA Signature Verification • HMAC-SHA1 Message Authentication • HMAC-SH256 Message Authentication • HMAC-SH384 Message Authentication • HMAC-SH512 Message Authentication • Resource Initialization/Cleanup • RSA Key Export • RSA Key Import • RSA Signature Generation • RSA Signature Verification • Self-tests • SHA-1 • SHA-256 • SHA-384 • SHA-512 • Show Status • Set Module Mode • TDES Encryption • TDES Decryption • TDES IV Import

	• TDES Key Import
Cryptographic Officer	• AES Encryption • AES Decryption • AES IV Import • AES Key Import • DH Key Generation • DH Key Exchange • DSA Key Export • DSA Key Generation • DSA Key Import • DSA Signature Generation • DSA Signature Verification • HMAC-SHA1 Message Authentication • HMAC-SH256 Message Authentication • HMAC-SH384 Message Authentication • HMAC-SH512 Message Authentication • Resource Initialization/Cleanup • RSA Key Export • RSA Key Import • RSA Signature Generation • RSA Signature Verification • Self-tests • SHA-1 • SHA-256 • SHA-384 • SHA-512 • Show Status • Set Module Mode • TDES Encryption • TDES Decryption • TDES IV Import • TDES Key Import

Table 8 Services Authorized for Roles

9.1 Definition of Critical Security Parameters (CSPs)

The following are CSPs contained in the computer's RAM used by the module (note, the term 'Externally' in the Generation column means that the CSP is provided by the calling application and is imported into the module via Import Services and APIs (including HMACSetup) listed in Table 7):

KEY	DESCRIPTION	GENERATION	STORAGE	ENTRY/OUTPUT	DESTRUCTION
DH Private	Used to derive	Internally using	Temporarily in	N/A	An application

Components	the secret session key during DH key agreement protocol	the DRNG (MS CAPIs)	volatile RAM		program which uses the API may destroy the key. The destruction service zeroes this CSP
RSA Private Key	Used to create RSA digital signatures.	Externally or through key generation routines. If through key generation routines, then they cannot be used in FIPS mode. RSA key generation (KeyGen) only operates in non-FIPS mode and all of the keys and and CSP's are zeroized when the module mode is changed (through unloading the module).	Temporarily in volatile RAM	Entry: Plaintext Output: N/A	An application program which uses the API may destroy the key. The Key Destruction service zeroes this CSP.
DSA Private Key	Used to create DSA digital signatures	Externally or through key generation routines	Temporarily in volatile RAM	Entry: Plaintext Output: N/A	An application program which uses the API may destroy the key. The Key Destruction service zeroes this CSP.
TDES Key	Used during TDES encryption and decryption	Externally	Temporarily in volatile RAM	Entry: Plaintext Output: N/A	An application program which uses the API may destroy the key. The Key Destruction service zeroes this CSP.
AES Key	Used during AES encryption and decryption.	Externally	Temporarily in volatile RAM	Entry: Plaintext Output: N/A	An application program which uses the API

					may destroy the key. The Key Destruction service zeroes this CSP.
HMAC Key	Used during HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384 or HMAC-SHA512 operation	Externally	Temporarily in volatile RAM	Entry: Plaintext Output: N/A	An application program which uses the API may destroy the key. The Key Destruction service zeroes this CSP.
Initialization Vectors	Used during AES/TDES cipher initialization	Externaly	Temporarily in volatile RAM	Entry: Plaintext Output: N/A	Provided to the module during cipher initialization. Destroyed when the cipher resource are freed via CipherFree

Table 9 CSP Information

The keys and CSPs saved by an application that loaded the Module DLL are stored outside of the Module memory and are the responsibility of the developer writing the application. The developer is directed to use HASHFree(), CIPHERFree(), HMACTearDown(), DHFree(), RSAFreeKeys(), and DSAFreeKeys() listed in Table 7 to zeroize and free these saved structures. Inside the Module proper, the keys and CSPs persist only in memory and only for the duration of each API's execution. If a CSP or key is maintained outside of the Microsoft CAPI's produced structures but within the Module's memory, then the memory is zeroed manually by the Module prior to the key or CSP being destroyed. Zeroization of the keys and CSPs contained in the Microsoft CAPI's produced structures is performed by the Module via the destruction APIs provided by the FIPS 140-2 validated Microsoft CAPI's.

9.2 Definition of Public Keys:

The following are the public keys contained in the module:

KEY	DESCRIPTION	GENERATION	STORAGE	ENTRY/OUTPUT
DH Public Component	Used to derive the secret session key during DH key agreement protocol	Internally using the FIPS 186-2 DRNG (MS CAPIs)	Temporarily in volatile RAM	Entry: Receive Client Public Component during DH exchange.

				Output: Host Public Component
RSA Public Keys	Used to verify RSA signatures	Externally or generated through RSA routines. If through key generation routines, then they cannot be used in FIPS mode. RSA key generation (KeyGen) only operates in non-FIPS mode and all of the keys and and CSP's are zeroized when the module mode is changed (through unloading the module).	Temporarily in volatile RAM	Input: Plaintext Output: N/A
DSA Public Keys	Used to verify DSA signatures	May be internally generated or generated externally.	Temporarily in volatile RAM	Input: Plaintext if generated externally Output: Plaintext

Table 10 Public Key Information

9.3 Definition of CSPs Modes of Access

The following table defines the relationship between access to CSPs and the different module services.

CRYPTO OFFICER ROLE	USER ROLE	SERVICE	CRYPTOGRAPHIC KEYS AND CSP VALUES ACCESS OPERATION
X	X	AES Encryption	Use AES Key (RE)
X	X	AES Decryption	Use AES Key (RE)
X	X	AES IV Import	AES Initialization Vector (W)
X	X	AES Key Import	AES Key (W)
X	X	DH Key Generation	Use DH Parameters (WE) Generate DH Key Pair (RE)
X	X	DH Key Exchange	Use DH Private Component (RE) Generate DH shared secret (RE)Generate DSA Signature (RE)
X	X	DSA Key Export	Export Public/Private DSA Keys (R)
X	X	DSA Key Generation	Generate DSA Public/Private Key (E)

X	X	DSA Key Import	Import Public/Private DSA Keys (W)
X	X	DSA Signature Generation	Use DSA Private Key (RE) Generate DSA Signature (RE)
X	X	DSA Signature Verification	Use DSA Public Key (RE) Verify DSA Signature (RE)
X	X	HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 Message Authentication Code	Import HMAC Key (RWE) Generate HMAC Output (RE)
X	X	Resource Initialization/Cleanup	Zero and Free: - AES Key - DH Private Components, - DSA Public/Private Keys - HMAC Key - Initialization Vectors (EW)
X	X	RSA Key Export	Export Public/Private RSA Keys (R)
X	X	RSA Key Import	Import Public/Private RSA Keys (W)
X	X	RSA Signature Generation	Use RSA Private Key (RE) Generate DSA Signature (RE)
X	X	RSA Signature Verification	Use RSA Public Key (RE) Verify RSA Signature (RE)
X	X	Self-tests	(E)
X	X	SHA-1, SHA-256, SHA-384, SHA-512	Generate Hash Output (RE)
X	X	Show Status	(R)
X	X	Set Module Mode	(W)
X	X	TDES Encryption	Use TDES Key (RE)
X	X	TDES Decryption	Use TDES Key (RE)
X	X	TDES IV Import	TDES Initialization Vector (W)
X	X	TDES Key Import	TDES Key (W)

Table 11 CSP Access Rights within Roles and Services

10 Operational Environment

When a crypto module is implemented in a server environment, the server application is the user of the cryptographic module. The server application makes the calls to the cryptographic module. Therefore, the server application is the single user of the cryptographic module, even when the server application is serving multiple clients.

The FIPS 140-2 Area 6 Operational Environment requirements are applicable because the Pragma Systems Cryptographic Module operates in a modifiable operational environment. The following Operational Environments are supported:

- Microsoft Windows 2003 Server
- Microsoft Windows 2008 Sever
- Microsoft Windows 2008 R2 Server
- Microsoft Windows Vista
- Microsoft Windows 7

For the purposes of FIPS 140-2 validation, the module was tested on the following platforms:

- Microsoft Windows 2003 Server
- Microsoft Windows 2008 Sever
- Microsoft Windows Vista

Pragma affirms that the module will also operate correctly on the following platforms, however no assurances are made by the CMVP regarding the correct operation of the module under these operational environments:

- Microsoft Windows 8.1
- Microsoft Windows Server 2012 R2

11 Physical Security

The FIPS 140-2 Area 5 Physical Security requirements are not applicable because the Pragma Systems Cryptographic Module is software only.

12 Mitigation of Other Attacks Policy

The module has not been designed to mitigate any specific attacks outside the scope of FIPS 140-2 requirements.

13 Cryptographic Officer Guidance

- There are no installation or configuration instructions required for the Pragma Systems Cryptographic Module except for the standard usage of the Module DLL, import library and header file to develop the applications using the module. RSA Key generation is not allowed in the FIPS Approved mode of operation. For Development environments, the Module DLL, import library and headerfiles should be protected using OS file

permissions so that only authorized personnel can use the files. For Operating environments, the Module DLL should be protected using OS file permissions so that only authorized personnel can use the files.