FIPS 140-2 Security Policy

SIEMENS PLM Software Teamcenter Cryptographic Module

SIEMENS PLM Software
5800 Granite Parkway, Suite 600
Plano, TX 75024
USA

April 18, 2016

Version 3.0

Based on OpenSSL

This product includes cryptographic software written by

Eric Young (eay@cryptsoft.com)
Tim Hudson (tjh@cryptsoft.com)

SIEMENS

Non-Proprietary and Unrestricted:
This document contains information that is non-proprietary to Siemens PLM Software Inc. and its use is unrestricted.

Table of Contents

FIPS 140-2 Non-Proprietary Security Policy

1. Introduction

The following describes the security policy for the SIEMENS PLM Software Teamcenter Cryptographic Module (**TCM**).

- The logical cryptographic boundary of the TCM is the **TcCryptoFips** library and it is a shared library.
- The TcCryptoFips library provides FIPS-validated encryption, hashing, digital signatures, and random number generation.
- The TcCryptoFips library provides a C-language application program interface (API) for use by applications that require cryptographic functionality. The library is classified by FIPS 140-2 as a software module, multi-chip standalone module embodiment.
- The physical cryptographic boundary is the general purpose computer on which the TCM is installed.
- The TcCryptoFips library performs no communications other than with the calling library namely **TcCrypto** library, responsible of invoking the TCM services in FIPS mode.
- The software (TcCryptoFips library) version for this validation is 3.0.
- The TCM requires an initialization sequence (see IG 9.5):
    - Upon load, the TcCryptoFips library runs the integrity test followed by the self-tests.
    - When the calling application requests the module to be in FIPS mode, the TCM invokes FIPS_mode_set()implemented in TcCryptoFips library:
        - Verifies the user password.
        - Re-runs the algorithms test then returns a “1” for success and “0” for failure. If FIPS_mode_set() fails then all cryptographic services in the FIPS module will fail from then on. The module may still be initialized in domestic mode later. The application can test to see if FIPS mode has been successfully performed.
- The TCM is a cryptographic engine library, which can be used only in conjunction with additional software. Aside from the use of the NIST defined elliptic curves as trusted third party domain parameters, all other FIPS 186-3 assurances are outside the scope of the TCM, and are the responsibility of the calling process.

The TCM meets the FIPS 140-2 requirements at an overall security level 1. The individual FIPS 140-2 security levels for the TCM are as follows:

Security Requirement	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	2
Finite State Model	1
Physical Security	NA
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	3
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	NA

Table 1 – Security Level of Security Requirements

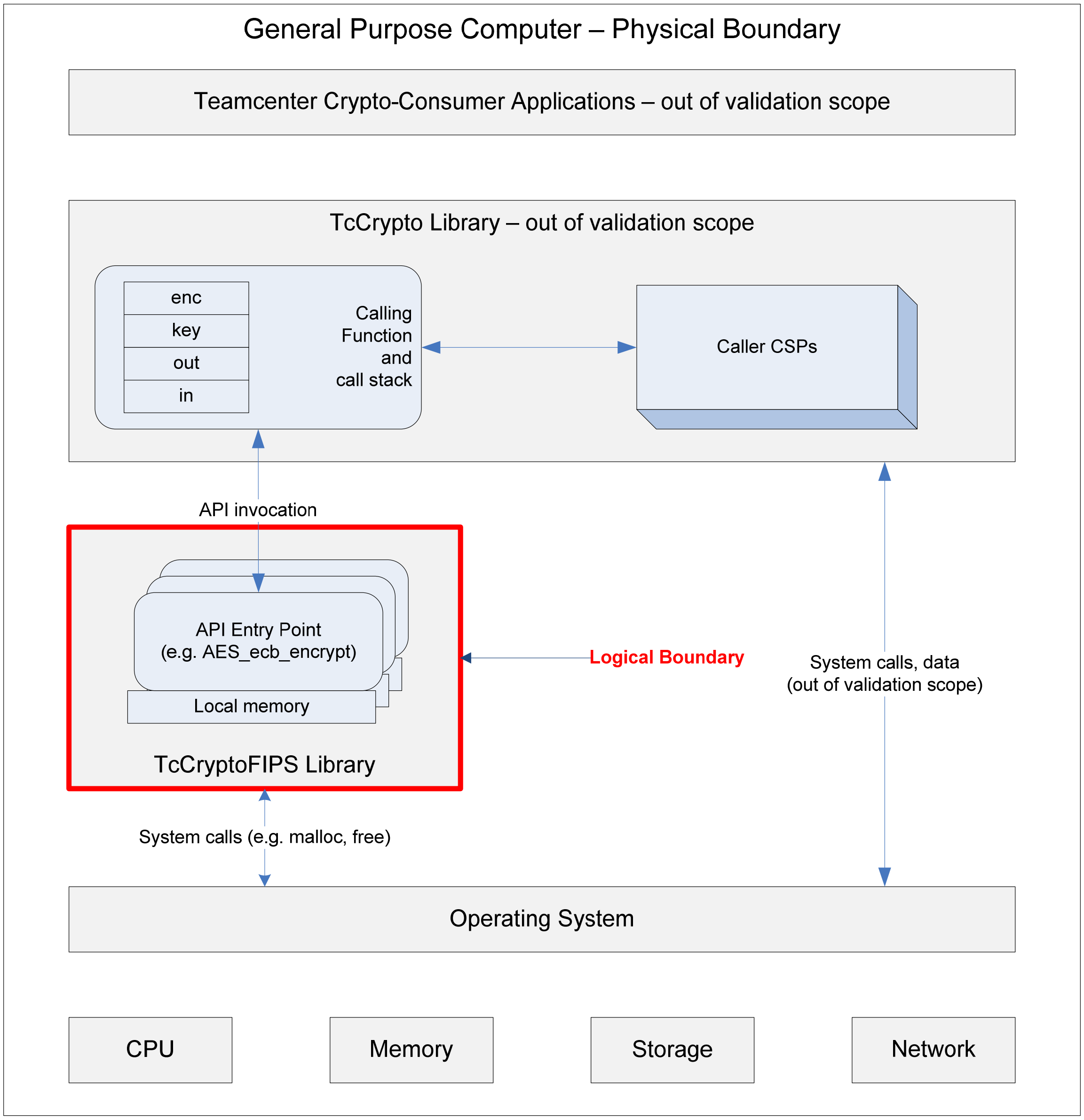

Figure 1 – TCM Block Diagram

1.1. Purpose

This document covers the secure operation of the TCM including the initialization, roles, and responsibilities of operating the product in a secure, FIPS-compliant manner.

1.2. References

Reference	Full Specification Name
OpenSSL	http://www.openssl.org/
[ANS X9.31]	Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)
[FIPS 140-2]	Security Requirements for Cryptographic modules, May 25, 2001
[FIPS 180-3]	Secure Hash Standard
[FIPS 186-4]	Digital Signature Standard
[FIPS 197]	Advanced Encryption Standard
[FIPS 198-1]	The Keyed-Hash Message Authentication Code (HMAC)
[SP 800-38B]	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication
[SP 800-38C]	Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality
[SP 800-38D]	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
[SP 800-56A]	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
[SP 800-67R1]	Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher
[SP 800-89]	Recommendation for Obtaining Assurances for Digital Signature Applications
[SP 800-90]	Recommendation for Random Number Generation Using Deterministic Random Bit Generators
[SP 800-131A]	Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths

Table 1.2 - References

1.3. Glossary

Term/Acronym	Description
TCM	Teamcenter Cryptographic Module
CO	Cryptographic-officer or Crypto-officer

Table 1.3 - Glossary

2. Ports and Interfaces

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

Logical interface type	Description
Control input	API entry point and corresponding stack parameters
Data input	API entry point data input stack parameters
Status output	API entry point return values and status stack parameters
Data output	API entry point data output stack parameters

Table 2 - Logical interfaces

As a software module, control of the physical ports is outside TCM scope. However, when the TcCryptoFips library is performing self-tests, or is in an error state, all output on the logical data output interface is inhibited. In error scenarios, the TcCryptoFips library returns only an error value (no data output is returned).

3. Modes of Operation and Cryptographic Functionality

3.1. Approved Mode

The TcCryptoFips library supports only a FIPS 140-2 Approved mode. Tables 3.1a and 3.1b list the Approved and Non-approved but Allowed algorithms, respectively.

Function	Algorithm	Options	Cert #
Random Number Generation; Symmetric key generation	[SP 800-90A] DRBG	Hash DRBG HMAC DRBG, no reseed CTR DRBG (AES), no derivation function	988
Encryption, Decryption and CMAC	[SP 800-67] Triple-DES	3-Key TRIPLE-DES TECB, TCBC TOFB; CMAC generate and verify	2058
	[FIPS 197] AES	128/192/256 ECB, CBC, OFB, CFB 1, CFB 8,CFB 128, CTR, CCM, GCM, CMAC generate and verify; 128/256 XTS;	3680
	[SP 800-38B] CMAC [SP 800-38C] CCM [SP 800-38D] GCM [SP 800-38E] XTS		3680
Message Digests	[FIPS 180-3] SHA	SHA-1, SHA-2 (224, 256, 384, 512)	3094
Keyed Hash	[FIPS 198] HMAC	SHA-1, SHA-2 (224, 256, 384, 512)	2426
Digital Signature and Asymmetric Key Generation	[FIPS 186-4] RSA	SigGen9.31, SigGenPKCS1.5, SigGenPSS, SigVer9.31, SigVerPKCS1.5,SigVerPSS	1901
	[FIPS 186-4] DSA	PQG Gen, PQG Ver, Key Pair Gen, Sig Gen, SigVer (2048/3072 with all SHA-2 sizes). PQG Ver and SigVer for 1024.	1037
	[FIPS 186-4] ECDSA	PKG: CURVES(P-224 P-256 P-384 P-521 ExtraRandomBits TestingCandidates) PKV: CURVES(ALL-P) SigGen: CURVES P-224: (SHA-224, 256, 384, 512) P-256: (SHA-224, 256, 384, 512) P-384: (SHA-224, 256, 384, 512) P-521: (SHA-224, 256, 384, 512)	774

		SigVer: CURVES P-192: (SHA-1, 224, 256, 384, 512) P-224: (SHA-1, 224, 256, 384, 512) P-256: (SHA-1, 224, 256, 384, 512) P-384: (SHA-1, 224, 256, 384, 512) P-521: (SHA-1, 224, 256, 384, 512)	
ECC CDH (CVL)	[SP 800-56A] KAS	All NIST defined P curves except sizes 163 and 192	676

Table 3.1a – FIPS Approved Cryptographic Functions

Notes:

- GCM is implemented in C with assembler language optimization for some platforms.
- The IV is generated internally to the cryptographic module. It is generated internally to the GCM algorithm boundary, assuming the algorithm boundary is the same as the Module boundary. The implementation is in the same C source file as other GCM and AES code.
- The IV fixed field size will have a minimum size of 4 bytes in approved mode. The contents are supplied by the caller (n.b.: this is required by TLS 1.2) based on the invocation.
- The IV fixed field contents allows for at least 2^32 different names.
- The IV invocation field has a minimum size of 64 bits in approved mode. The contents are initially from an approved DRBG source, with the alternative of all zeros or a value supplied by the caller. It increments by 1.
- It will take 2^64 increments for the IV invocation field to wrap. The module does not enter an error state if wrapping occurs because it is inconceivable that this value can wrap around.
- Assuming a time of 1ns per generation operation (several orders of magnitude faster than currently possible) it would take over 584 years to wrap around.
- If Module power is lost and restored, the caller can set the IV to the last value used.
- EC DH Key Agreement provides a minimum of 112 bits and a maximum of 256 bits of security strength as indicated in Table 2 of NIST publication, SP800-57
- RSA key wrapping provides a minimum of 112 bits and a maximum of 150 bits of security strength as indicated in Table 2 of NIST publication, SP800-57
- The TcCryptoFips library supports only NIST defined curves for use with ECDSA and ECC CDH. The TcCryptoFips library supports only one operational environment configuration for elliptic curve; NIST prime curve only (listed in Table 2 with the EC column marked "P").

Category	**Algorithm**	**Description**
Key Agreement	EC DH	Non-compliant (untested) DH scheme using elliptic curve, supporting all NIST defined P curves. Key agreement is a service provided for calling process use, but is not used to establish keys into the TCM.
Key Encryption, Decryption	RSA	The RSA algorithm may be used by the calling application for encryption or decryption of keys. No claim is made for SP 800-56B compliance, and no CSPs are established into or exported out of the TCM using these services.
Random Number Generation	NDRNG	Entropy data is required to seed the DRBG. Several entropy sources are used by the library to collect entropy on different supported platforms. RAND_poll function is the entropy gathering function that uses sources for example, on Windows platform, it uses sources such as network data, random data from cryptographic service provider using CryptGenRandom, kernel-based entropy data using heap status entries and the state of global memory. On Unix platforms it reads /dev/random as a random entropy pool device.

Table 3.1b – Non-FIPS Approved But Allowed Cryptographic Functions

3.2. Non-Approved (non-FIPS mode only)

The following algorithms shall not be used when operating in the FIPS Approved mode of operation:

- MD5
- DES
- HMAC (key length < 112 bits)
- Diffie-Hellman, key sizes supported :1024 and 2048
- Random Number Generation [ANS X9.31] RNG using AES 128/192/256 cert# 1408
- FIPS 186-2 RSA, GenKey9.31, SigGen9.31, SigGenPKCS1.5, SigGenPSS (1024/1536 with all SHA sizes, 2048/3072/4096 with SHA-1)
- FIPS 186-2 DSA, PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)
- FIPS 186-2 ECDSA, PKG: CURVES (P-192), SIG(gen): CURVES (P-192 P-224 P-256 P-384 P-521)

3.3. Critical Security Parameters and Public Keys

All CSPs used by the TCM are described in this section. All access to these CSPs by TCM services are described in Section 3. The CSP names are generic, corresponding to API parameter data structures.

CSP Name	Description
RSA SGK	RSA (2048, 3072-bit) signature generation key
RSA KDK	RSA (1024, 2048, 3072, and 4096-bit) key decryption (private key transport) key
DSA SGK	[FIPS 186-4] DSA (2048/3072) signature generation key
ECDSA SGK	ECDSA (All NIST defined P curves) signature generation key
EC DH Private	EC DH (All NIST defined P curves) private key agreement key.
AES EDK	AES (128/192/256) encrypt / decrypt key
AES CMAC	AES (128/192/256) CMAC generate / verify key
AES GCM	AES (128/192/256) encrypt / decrypt / generate / verify key
AES XTS	AES (256/512) XTS encrypt / decrypt key
TRIPLE-DES EDK	TRIPLE-DES (3-Key) encrypt / decrypt key
TRIPLE-DES CMAC	TRIPLE-DES (3-Key) CMAC generate / verify key
HMAC Key	Keyed hash key (160/224/256/384/512)
Hash_DRBG CSPs	V (440/888 bits) and C (440/888 bits), entropy input (length dependent on security strength)
HMAC_DRBG CSPs	V (160/224/256/384/512 bits) and Key (160/224/256/384/512 bits), entropy input (length dependent on security strength)
CTR_DRBG CSPs	V (128 bits) and Key (AES 128/192/256), entropy input (length dependent on security strength)
CO-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for Crypto Officer role authentication
User-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for User role authentication

Table 3.3 – Critical Security Parameters

For all CSPs and Public Keys:

Storage: RAM, associated to entities by memory location. The TCM stores DRBG and DRBG state values for the lifetime of the DRBG instance. The TCM uses CSPs passed in by the calling application on the stack. The TCM does not store any CSP persistently (beyond the lifetime of an API call), with the exception of DRBG and DRBG state values used for the TCMs' default key generation service.

The crypto officer and user authorization hash data is persistently stored in a read only segment of the TcCryptoFips library. The user authorization data is stored by the calling entity such as in our case, the read only segment of TcCrypto module. The crypto officer authorization data is stored by the calling entity such as in our case, the read only segment of an unpublished utility application.
Generation: SP 800-90A compliant DRBG services for creation of symmetric keys, and for generation of DSA, elliptic curve as shown in Table 3.3. The calling application is responsible for storage of generated keys returned by the TCM.
Entry: All CSPs enter the TCM's logical boundary in plaintext as API parameters, associated by memory location. However, none cross the physical boundary.
Output: The TCM does not output CSPs, other than as explicit results of key generation services. However, none cross the physical boundary.
Destruction: Zeroization of sensitive data is performed automatically by API function calls for temporarily stored CSPs. In addition, the TCM provides functions to explicitly destroy CSPs related to random number generation services. The calling application is responsible for parameters passed in and out of the TCM.

- **Private and secret keys** as well as seeds and entropy input are provided to the TCM by the calling application, and are destroyed when released by the appropriate API function calls. When entropy is externally loaded, no assurance of the minimum strength of generated keys.

- **Keys residing in internally allocated data structures** (during the lifetime of an API call) can only be accessed using the TCM defined API. The operating system protects memory and process space from unauthorized access. Only the calling application that creates or imports keys can use or export such keys. All API functions are executed by the invoking calling application in a non-overlapping sequence such that no two API functions will execute concurrently.

- **In the event TCM power is lost** and restored the calling application must ensure that any AES-GCM keys used for encryption or decryption are re-distributed.

- **TcCryptoFips library users (the calling applications) shall use entropy sources**, as described in section 4, that meet the security strength required for the random number generation mechanism: [SP 800-90] Table 2 (Hash_DRBG, HMAC_DRBG), Table 3 (CTR_DRBG). This entropy is supplied by means of callback functions. Those functions must return an error if the minimum entropy strength cannot be met.

4. Compliant DRBG Entropy Source

There are multiple cryptographic functions supported by the TCM library that rely on the availability of random numbers. Generating symmetric session keys and public/private key pairs are some of these functions.
To meet this requirement, the TCM provides a cryptographically strong DRBG so that the random data is computationally hard to predict. The DRBG require a secret seed that is high in entropy to set its initial state to ensure that its output will not be determined.

The following entropy collection procedure is used to seed the FIPS compliant DRBG:

- RAND_add is called and the collected entropy data is passed in as a function parameter.

- RAND_get_rand_method returns the addresses of the set method in a structure of type RAND_METHOD which holds pointers to the functions.

- RAND_add() calls meth->add() which points to the physical function ssleay_rand_add where the collected entropy data is hashed directly into the DRBG's state.

Refer to the Figure 2 for a diagram that illustrates all of the components, sources and mechanisms that constitute an entropy source used by the TCM's DRBG.

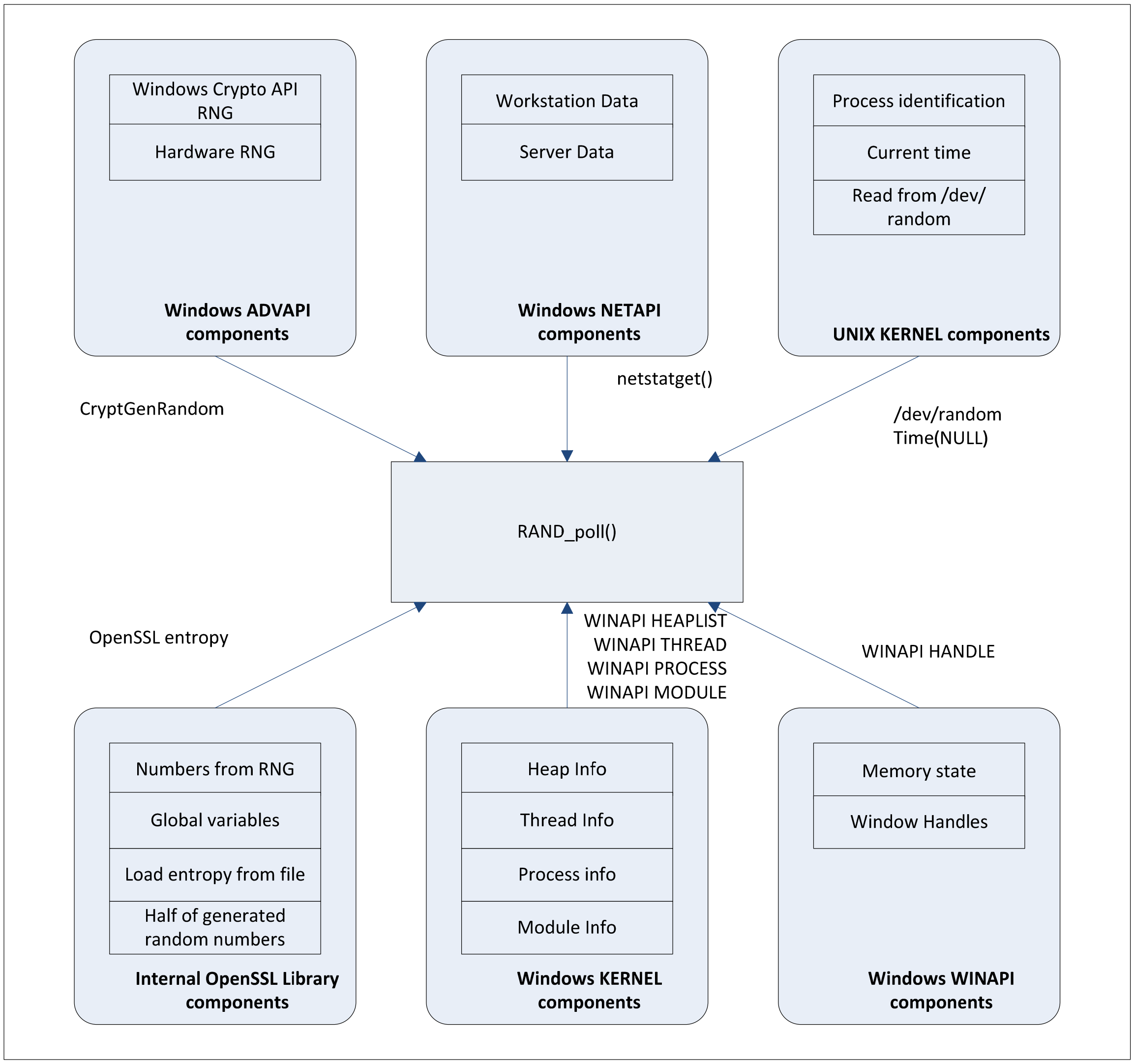

Figure 2 – TCM entropy sources and components

4.1. Entropy collection process

Entropy data is collected and added into the DRBG's entropy pool. RAND_poll function is the function responsible of entropy collection. Entropy data sources are depicted in Figure 2.

Windows Platform

1. Collect network data:
   Uses netstatget() returns a STAT_WORKSTATION_0 structure of 45 fields of data and STAT_SERVER_0 structure of another 17 fields. Each field is estimated as 1-byte of entropy to be passed to RAND_add.
2. Collect random data from cryptographic service:

Uses the cryptographic service provider in hProvider to call CryptGenRandom and obtain 64 bytes of random data from processor's on-chip DRBG. If successful, 64 bytes of random data are passed to RAND_add.

3. Get kernel-based entropy data:
   Using the heap status and collecting entropy from each entry, the process list, thread list and module list.
4. Add the state of global physical and virtual memory:
   The current process ID is added to ensure that each thread has something different than the others.
5. Load Entropy from file:
   Additional entropy is added by calling RAND_load_file function which adds the file content using RAND_add.

UNIX Platforms

1. Read 32 bytes from /dev/random or /dev/random to be passed RAND_add.
2. Get current time using time(NULL) to be passed to RAND_add.

5. Roles, Services, and Authentication

The TcCryptoFips library provides two roles and sets of services. The TcCryptoFips library starts up with an application calling an initialize function, and then provides cryptographic capabilities on behalf of the user.

5.1. Roles

All operations occur on behalf of the application running operations for the user of the application software.

The TcCryptoFips library supports both User and Crypto-officer roles. The User role has access to all services of the TCM.

- User Role (User): Loading the TCM, setting the TCM in FIPS mode, and calling any of the FIPS API functions.
- Crypto Officer Role (CO): Installation of the TCM on the host computer system and calculating the signature code used by the integrity check.

5.2. Services

All services implemented by the TCM are listed below in Table-5.2a, along with a description of service CSP access and the API function call.

Service	Role	Description
Initialize	User, CO	TCM initialization. Does not access CSPs.
Self-test	User	Perform self tests (FIPS_selftest). Does not access CSPs.
Show status	User	Functions that provide TCM status information:

		• Version (as unsigned int or const char *) • FIPS Mode (Boolean) Does not access CSPs.
Zeroize	User	Functions that destroy CSPs: • fips_drbg_uninstantiate: for a given DRBG context, overwrites DRBG CSPs (Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs.) All other services automatically overwrite CSPs stored in allocated memory. Stack cleanup is the responsibility of the calling application.
Random number generation	User	Used for random number and symmetric key generation. • Seed or reseed DRBG instance • Determine security strength of a DRBG instance • Obtain random data Uses and updates DRBG CSPs, Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs. FIPS_drbg_free FIPS_drbg_generate FIPS_drbg_get_app_data FIPS_drbg_get_blocklength FIPS_drbg_get_strength FIPS_drbg_health_check FIPS_drbg_init FIPS_drbg_instantiate FIPS_drbg_method FIPS_drbg_new FIPS_drbg_reseed FIPS_drbg_set_app_data FIPS_drbg_set_callbacks FIPS_drbg_set_check_interval FIPS_drbg_set_rand_callbacks FIPS_drbg_set_reseed_interval FIPS_drbg_stick FIPS_drbg_uninstantiate
Asymmetric key generation	User	Used to generate DSA and ECDSA: DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK There is one supported entropy strength for each mechanism and algorithm type, the maximum specified in SP800-90 FIPS_dsa_generate_key FIPS_dsa_generate_parameters_ex

		FIPS_ec_group_get_degree FIPS_ec_group_method_of FIPS_ec_group_new_by_curve_name FIPS_ec_group_new_curve_gfp FIPS_ec_key_free FIPS_ec_key_generate_key FIPS_ec_key_get0_group FIPS_ec_key_get0_private_key FIPS_ec_key_get0_public_key FIPS_ec_key_new FIPS_ec_key_new_by_curve_name FIPS_ec_key_set_flags FIPS_ec_key_set_group FIPS_ec_key_set_private_key FIPS_ec_key_set_public_key_affine_coordinates FIPS_ec_method_get_field_type FIPS_ec_point_free FIPS_ec_point_get_affine_coordinates_gfp FIPS_ec_point_new
Symmetric encrypt/decrypt	User	Used to encrypt or decrypt data. Executes using AES EDK, TRIPLE-DES EDK (passed in by the calling process). FIPS_evp_aes_128_cbc FIPS_evp_aes_128_ccm FIPS_evp_aes_128_cfb1 FIPS_evp_aes_128_cfb128 FIPS_evp_aes_128_cfb8 FIPS_evp_aes_128_ctr FIPS_evp_aes_128_ecb FIPS_evp_aes_128_gcm FIPS_evp_aes_128_ofb FIPS_evp_aes_128_xts FIPS_evp_aes_192_cbc FIPS_evp_aes_192_ccm FIPS_evp_aes_192_cfb1 FIPS_evp_aes_192_cfb128 FIPS_evp_aes_192_cfb8 FIPS_evp_aes_192_ctr FIPS_evp_aes_192_ecb FIPS_evp_aes_192_gcm FIPS_evp_aes_192_ofb FIPS_evp_aes_256_cbc FIPS_evp_aes_256_ccm

		FIPS_evp_aes_256_cfb1 FIPS_evp_aes_256_cfb128 FIPS_evp_aes_256_cfb8 FIPS_evp_aes_256_ctr FIPS_evp_aes_256_ecb FIPS_evp_aes_256_gcm FIPS_evp_aes_256_ofb FIPS_evp_aes_256_xts FIPS_evp_des_ede3 FIPS_evp_des_ede3_cbc FIPS_evp_des_ede3_cfb1 FIPS_evp_des_ede3_cfb64 FIPS_evp_des_ede3_cfb8 FIPS_evp_des_ede3_ecb FIPS_evp_des_ede3_ofb
Symmetric digest	User, CO	Used to generate or verify data integrity with CMAC. Executes using AES CMAC, TRIPLE-DES, CMAC (passed in by the calling process). FIPS_cmac_ctx_cleanup; FIPS_cmac_ctx_free; FIPS_cmac_ctx_new; FIPS_cmac_final; FIPS_cmac_init; FIPS_cmac_update;
Message digest	User, CO	Used to generate a SHA-1 or SHA-2 message digest. Does not access CSPs. FIPS_digest FIPS_digestfinal FIPS_digestinit FIPS_digestupdate
Keyed Hash	User, CO	Used to generate or verify data integrity with HMAC. Executes using HMAC Key (passed in by the calling process). FIPS_hmac FIPS_hmac_ctx_cleanup FIPS_hmac_ctx_copy FIPS_hmac_ctx_init FIPS_hmac_ctx_set_flags FIPS_hmac_final FIPS_hmac_init FIPS_hmac_init_ex FIPS_hmac_update
Key transport	User	Used to encrypt or decrypt a key value on behalf

		of the calling process (does not establish keys into the TCM). Executes using RSA KDK, RSA KEK (passed in by the calling process). FIPS_rsa_private_decrypt FIPS_rsa_private_encrypt FIPS_rsa_public_decrypt FIPS_rsa_public_encrypt
Key agreement	User	Used to perform key agreement primitives on behalf of the calling process (does not establish keys into the TCM). Executes using EC DH Private, EC DH Public (passed in by the calling process). FIPS_ecdh_compute_key
Digital signature	User	Used to generate or verify RSA, DSA or ECDSA digital signatures. Executes using RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK (passed in by the calling process). FIPS_ecdsa_openssl FIPS_ecdsa_sig_free FIPS_ecdsa_sign FIPS_rsa_sign FIPS_rsa_sign_ctx FIPS_rsa_sign_digest FIPS_rsa_verify FIPS_rsa_verify_ctx
Utility	User, CO	Miscellaneous helper functions. Does not access CSPs. FIPS_cipher FIPS_cipher_ctx_cleanup FIPS_cipher_ctx_copy FIPS_cipher_ctx_ctrl FIPS_cipher_ctx_free FIPS_cipher_ctx_init FIPS_cipher_ctx_new FIPS_cipher_ctx_set_key_length FIPS_cipherinit fips_cipher_test FIPS_check_incore_fingerprint FIPS_bn_bin2bn FIPS_bn_bn2bin FIPS_bn_clear_free FIPS_bn_free FIPS_bn_is_prime_ex FIPS_bn_new FIPS_bn_num_bits

Table 5.2a - Services and CSP Access (Approved Mode)

List of all APIs implemented by the TCM in the "Non approved API's" service is listed below:

Service	**Role**	**Description**
Zeroize	User	Functions that destroy CSPs: • fips_rand_prng_reset: destroys RNG CSPs. All other services automatically overwrite CSPs stored in allocated memory. Stack cleanup is the responsibility of the calling application.
Random number generation	User	Used for random number and symmetric key generation. Seed or reseed RNG instance Determine security strength of a RNG instance Obtain random data FIPS_x931_bytes FIPS_x931_method FIPS_x931_reset FIPS_x931_seed FIPS_x931_set_dt FIPS_x931_set_key FIPS_x931_status FIPS_x931_stick FIPS_x931_test_mode
Symmetric encrypt/decrypt	User	Used to encrypt or decrypt data. Executes using DES EDK (passed in by the calling process). FIPS_evp_des_ede FIPS_evp_des_ede_cbc FIPS_evp_des_ede_cfb64 FIPS_evp_des_ede_ecb FIPS_evp_des_ede_ofb
Message digest	User	Used to generate a MD5 message digest. Does not access CSPs. FIPS_digest FIPS_digestfinal FIPS_digestinit FIPS_digestupdate
Keyed Hash	User	Used to generate or verify data integrity with HMAC. Executes using HMAC Key <112-bits (passed in by the calling process). FIPS_hmac FIPS_hmac_ctx_cleanup

		FIPS_hmac_ctx_copy FIPS_hmac_ctx_init FIPS_hmac_ctx_set_flags FIPS_hmac_final FIPS_hmac_init FIPS_hmac_init_ex FIPS_hmac_update
Key agreement	User	Used to perform key agreement primitives on behalf of the calling process (does not establish keys into the TCM). Executes using DH Private, DH Public (passed in by the calling process). FIPS_dh_compute_key_padded

Table 5.2b - Services and CSP Access (Non-Approved Mode)

5.3. Authentication Mechanisms and Strength

The TCM implements the required User and Crypto Officer roles and requires authentication for those roles. Only one role may be active at a time and the TCM does not allow concurrent operators.

The User role is assumed by passing the appropriate password to the TcCryptoFipsGetAPI function.

The Crypto officer role is assumed by passing the appropriate password to the utility function that has access to the maintenance API only.

The password values may be specified at build time and must have a minimum length of 16 characters. Any attempt to authenticate with an invalid password will result in an immediate and permanent failure condition rendering the TCM unable to enter the FIPS mode of operation, even with subsequent use of a correct password.

Authentication data is loaded into the TCM during the TCM build process, performed by the Crypto Officer, and otherwise cannot be accessed.

Since minimum password length is 16 characters, the probability of a random successful authentication attempt in one try is a maximum of $1/256^{16}$, or less than $1/10^{38}$. The TCM permanently disables further authentication attempts after a single failure, so this probability is independent of time.

6. Secure Operation and Security Rules

In order to operate the Teamcenter Cryptographic Module securely, the operator should be aware of the security rules enforced by the TcCryptoFips library and should adhere to the physical security rules and secure operation rules required.

6.1. Security Rules

The security rules enforced by the TCM, result from the security requirements of FIPS 140-2.

FIPS 140-2 Security Rules

The following are security rules needed to operate the TCM securely, that stem from the requirements of FIPS PUB 140-2. The TCM enforces these requirements when initialized into FIPS mode.

1. When initialized to operate in FIPS mode, the TCM shall only use FIPS-approved cryptographic algorithms.
2. The replacement or modification of the TCM by unauthorized intruders is prohibited.
3. The Operating System enforces authentication method(s) to prevent unauthorized access to TCM services.
4. All Critical Security Parameters are verified as correct and are securely generated, stored, and destroyed.
5. All host system components that can contain sensitive cryptographic data (main memory, system bus, disk storage) must be located in a secure environment.
6. The referencing application accessing the TCM runs in a separate virtual address space with a separate copy of the executable code.
7. The unauthorized reading, writing, or modification of the address space of the TCM is prohibited.
8. The writable memory areas of the TCM (data and stack segments) are accessible only by a single application so that the TCM is in "single user" mode, i.e. only the one application has access to that instance of the TCM.
9. The operating system is responsible for multitasking operations so that other processes cannot access the address space of the process containing the TCM.

6.2. Secure Operation Initialization Rules

Because FIPS 140-2 prohibits the use of non-FIPS approved algorithms while operating in a FIPS compliant manner, the TCM should be initialized to ensure FIPS 140-2 level 1 compliance.

1. Start a Teamcenter application that uses the TCM.
2. When the TCM enters the Uninitialized state, it should call FIPS_mode_set() implemented in TcCryptoFips.
3. The application should check the return code to ensure the application initialization was successful.
4. When initialized in this fashion, the TCM will only use FIPS-approved algorithms. Note that the state of an TCM can be determined at any time by calling the FIPS_module_mode() function, which will return 1 if the TCM is operating in FIPS mode.

6.3. Operating Systems

The TCM has been officially validated on the following platforms and no claim can be made as to the correct operation of the TCM or the security strengths of the generated keys when operating on a platform that is not listed on the validation certificate:

- Windows 7 (x86 / x64)
  - Visual Studio 2013 (VC 12)
- Linux SuSE 11.2 (x64)
  - Compiler – g++ 4.3.4
- Mac OSX 10.11 (x64)
  - Compiler - clang LLVM version 7.0

In addition to the validation, the TCM has been tested by SIEMENS PLM on the following platforms:

- Windows
  - Windows 8 (x64 / x86) – Visual Studio 2015
  - Windows 8 (x64 / x86) – Visual Studio 2012
  - Windows 7 (x64 / x86) – Visual Studio 2010

- Linux
  - CentOS 6.x (x64 / x86) – g++
  - RedHat 6.x (x64/x86) – g++ 4.4.4
  - SUSE 10.x (x64 / x86) – g++ 4.1.2
- Solaris
  - Solaris 10 (32-bit / 64-bit) – Solaris Studio 12.3 C++ 5.12
- Mac OS X
  - OSX 10.8 (x64) – clang LLVM 4.2
  - OSX 10.10 (x64) – clang LLVM 6.1

- AIX
  - AIX 6.0 (32-bit / 64-bit) – xlC 11.1

7. Self-tests

The following list shows all self-tests implemented in the TcCryptoFips library.

In addition to these self-tests, the TcCryptoFips library also contains an embedded watermark that will be verified at runtime to ensure that the library has not been corrupted or modified.
Also, the library performs continuous Random Number Generator tests on the output of the Approved DRBG to ensure that it is not "stuck".

The TcCryptoFips library performs self-tests listed below on invocation of Initialize or Self-test.

Algorithm	Type	Test Attributes
Software integrity	KAT	HMAC-SHA512
HMAC	KAT	One KAT per SHA1, SHA224, SHA256, SHA384 and SHA512. This testing covers SHA POST requirements.
AES	KAT	Separate encrypt and decrypt, ECB mode, 128 bit key length
AES CCM	KAT	Separate encrypt and decrypt, 192 key length
AES GCM	KAT	Separate encrypt and decrypt, 256 key length
XTS-AES	KAT	128, 256 bit key sizes to support either the 256-bit key size (for XTS-AES-128) or the 512-bit key size (for XTS-AES-256)
AES CMAC	KAT	Sign and verify CBC mode, 128, 192, 256 key lengths
TRIPLE-DES	KAT	Separate encrypt and decrypt, ECB mode, 3-Key
TRIPLE-DES CMAC	KAT	CMAC generate and verify, CBC mode, 3-Key
RSA	KAT	Sign and verify using 2048 bit key, SHA-256, PKCS#1
DSA	PCT	Sign and verify using 2048 bit key, SHA-384
DRBG	KAT	CTR_DRBG: AES, 256 bit with and without derivation function HASH_DRBG: SHA256 HMAC_DRBG: SHA256

ECDSA	PCT	Keygen, sign, verify using P-224 and SHA512.
ECC CDH	KAT	Shared secret calculation per SP 800-56A §5.7.1.2, IG 9.6
X9.31 RNG	KAT	128, 192, 256 bit AES keys

Table 7a - Power On Self Tests (KAT = Known answer test; PCT = Pairwise consistency test)

The TCM also implements the following conditional tests:

Algorithm	**Test**
DRBG	Tested as required by [SP800-90A]
DRBG	FIPS 140-2 continuous test for stuck fault
NDRNG	FIPS 140-2 continuous test for stuck entropy
DSA	Pairwise consistency test on each generation of a key pair
ECDSA	Pairwise consistency test on each generation of a key pair
RSA	Pairwise consistency test on each generation of a key pair
ANSI X9.31 RNG	Continuous test for stuck fault

Table 7b - Conditional Tests

8. Mitigation of Other Attacks

This section is not applicable.