ZTE Unified Element Management Platform Cryptographic Module Version 4.11.10

FIPS 140-2 Non-Proprietary Security Policy
Policy Version 1.1
Last Updated: 2011-07-05

Table of contents

1. Introduction

This document is a non-proprietary FIPS 140-2 Security Policy for the Unified Element Management Platform Cryptographic Module ("UEP Cryptographic Module" or "UEPCM" or "module") version 4.11.10. It describes how this module meets the requirements as specified in FIPS PUB 140-2 (Federal Information Processing Standards Publication 140-2) for a Security Level 1 multi-chip standalone software module. The companion document ZTE UEPCM User's Guide is a technical reference for the UEPCM application developers to use and integrate the UEPCM into their applications.

The security policy is required for FIPS 140-2 validation and is intended to be part of the package of documents that are submitted to Cryptographic Module Validation Program (CMVP). It describes the capabilities, protection, and access rights provided by the cryptographic module. It also contains a specification of the rules under which the module must operate in order to be in the FIPS mode. This security policy allows individuals and organizations to determine whether the cryptographic module meet their security requirements and to determine whether the module, as implemented, satisfies the stated security policy.

The targeted audience of this document consists of but not limited to the UEPCM module and its application developers, testers at the Cryptographic Services Testing (CST) lab and reviewers from CMVP.

2.Cryptographic Module Specification

2.1. Module Overview

The Unified Element Management Platform Cryptographic Module (Software Version 4.11.10) is a software only, multi-chip standalone cryptographic module that runs on a blade system. The UEPCM is used by the ZTE to provide a solution for mutual authentication and cryptographic services between network elements and their management platform.

2.2. Cryptographic Boundary

The logical cryptographic boundary for the Module is the library itself. The format of this library is a jar file (uepcm.jar).

The physical boundary of the UEPCM is a blade in a blade system on which the module is to be executed. A blade is the hardware platform on which the module runs. A blade consists of standard integrated circuits, including processors, memory, hard disk, data I/O interface, clock and power input.

The block diagram for the module is shown below.

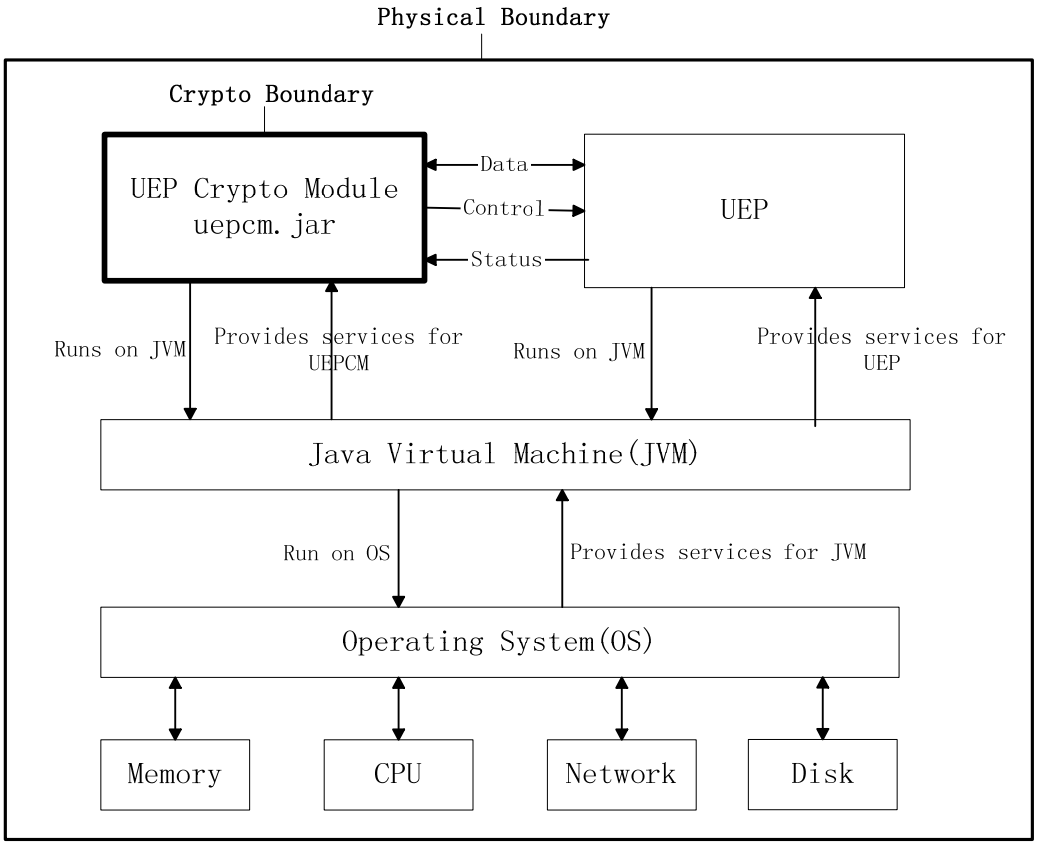

Figure 1 : Block Diagram of the Cryptographic Module

The UEPCM cryptographic module is developed in JAVA language and is provided to application developers as a JAR file. Applications interact with the cryptographic module through an API (see appendix A).

The relationship between the UEPCM and the ZTE UEP application is shown in the following diagram.

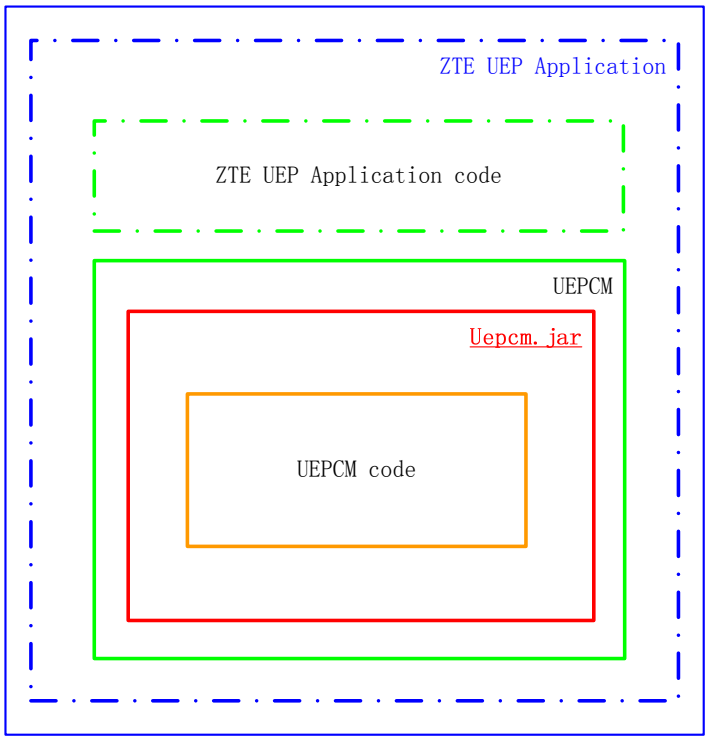

Figure 2: Functional Decomposition of the UEPCM and Its UEP Application

- ZTE UEP Application – The ZTE UEP application using the UEPCM. The UEP application is built upon the application code and the UEPCM JAR file.
- ZTE UEP Application code – The program using the UEPCM to perform cryptographic functions.
- UEPCM – The cryptographic module to be validated that contains a uepcm.jar that can be linked to the ZTE UEP application.
- uepcm.jar – a dynamically linkable cryptographic library that is compiled from the UEPCM code.

The module components within the logical boundary of the UEPCM are specified in Table 1.

Component Type	File Name	Version
Dynamic Library binary code	uepcm.jar	4.11.10
Documentation	UEPCM_SP.doc	1.0
	FiniteStateMachine(UEPCM).doc	1.1
	ComponentsList(UEPCM).doc	1.2

Table 1: The UEPCM Cryptographic Module Components

2.3. Cryptographic Module Security Level

The module is validated as a software module running on a multi-chip standalone platform against FIPS 140-2 at overall Security Level 1 cryptographic module. The following table shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2:

FIPS 140-2 Sections	Security Level
Cryptographic Module Specification	1
Cryptographic 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 - Security Levels for Eleven Sections of the FIPS 140-2 Requirements

2.4. Tested Platforms

The UEPCM validation was performed on the following platforms.

Manufacturer	Model	CPU	EMI/EMC Compliance	OS and Version	JDK/JRE Version
ZTE	SBCO	AMD Opteron® 64	FCC report provided by ZTE CORPORATION EMC Laboratory	NewStart CGS Linux V3.02 (single user mode)	Sun JDK/JRE 1.6.0_11
ZTE	SBCW	Intel® Xeon™ 64	FCC report provided by ZTE CORPORATION EMC Laboratory	NewStart CGS Linux V3.02 (single user mode)	Sun JDK/JRE 1.6.0_11

HP	BL680c Blade Server	Intel® Xeon™ 64	Conformance declaration provided by Hewlett-Packard Company	NewStart CGS Linux V3.02 (single user mode)	Sun JDK/JRE 1.6.0_11

Table 3 - Tested platforms

2.5. Approved Mode of Operation

The UEPCM module implements a list of FIPS-Approved algorithms and callable in FIPS-mode as shown in Table 4.

Algorithm w/modes	Keys/CSPs	Usage	Algorithm Certificate #
AES (ECB, CBC, OFB, CFB8, CFB128)	128, 192, 256 bit keys	Encryption/decryption	INTEL:# 1584 AMD: # 1583
Triple-DES (ECB, CBC, CFB8，CFB64, OFB)	3-key 168 bits	Encryption/decryption	INTEL:# 1040 AMD: # 1039
DSA (FIPS 186-3, PQG(gen), PQG(ver), KEYGEN, SIG(gen), SIG(ver))	p, q, g; 2048 bits modulus size 2048 bits modulus size key pair	Key Pair generation, Generate and Verify signature	INTEL:# 490 AMD: # 489
RSA (ANSI X9.31 (Key(gen));PSS (SIG(gen), SIG(ver)))	Module sizes: 1536, 2048, 3072, 4096; Public Key sizes: 3, 17, 65537	Key Pair generation, Generate and Verify signature	INTEL:# 774 AMD: # 773
DRBG SP800-90 (Hash_DRBG SHA256）	seed value and 256 bit seed key values.	Random bit generation.	INTEL:# 74 AMD: # 73
SHA224 (BYTE-only) SHA256 (BYTE-only) SHA384 (BYTE-only) SHA512 (BYTE-only)	N/A	Hashing	INTEL:# 1403 AMD: # 1402
HMACSHA-224 HMACSHA-256 HMACSHA-384 HMACSHA-512	HMAC keys Key Size Ranges Tested: Key Size < Block Size,	Message Authentication	INTEL:#930 AMD: # 929

	Key Size = Block Size, Key Size > Block Size		

Table 4: FIPS-Approved Cryptographic Algorithms

The UEPCM always operates in a FIPS approved mode of operation. For the purposes of FIPS- 140- 2 validation, the Module is classified as a multi-chip stand-alone Module.

3. Cryptographic Module Ports and Interfaces

As a software only cryptographic module, the logical interface of the UEPCM is its API documented in the ZTE UEPCM User's Guide. The execution of the UEPCM is powered by its hardware platform that connects to a power supply through its physical boundary. The table below maps the FIPS 140-2 required ports and interfaces to the UEPCM logical interfaces originated in its API.

FIPS Required Interface	The UEPCM Interface
Data Input	API input parameters
Data Output	API output parameters
Control Input	API function calls, API input parameters
Status Output	API return codes/messages
Power Input	Power connector through the underlying hardware power supply port

Table 5: Ports and Interface of the UEPCM

The UEPCM communicates any error status synchronously through the use of its documented return codes. It is optimized for library use and does not contain any terminating assertions or exceptions. The UEPCM does not support cryptographic bypass capability, either.

Any internal error detected by the UEPCM will be reflected back to the application with an appropriate return code or error message. The calling application must examine the return code and handle exceptional conditions in a FIPS 140-2 appropriate manner. Return codes and error conditions are fully documented in the ZTE UEPCM User's Guide. They do not reveal any sensitive material to callers.

4. Roles, Services and Authentication

4.1. Roles

The UEPCM supports two roles: a Cryptographic Officer role and a User role. Each of roles is authenticated through the operating system prior to using any system services. The module does not require any further authentication that would allow the module to distinguish between the two supported roles.

The Crypto Officer and User roles are implicitly assumed by the entity accessing services implemented by the module. Both roles can access all of the cryptographic services provided by the module. In addition, the Crypto Officer role can integrate the module with its applications, install and initialize the module. If any of the Crypto-Officer-specific services are involved, then the operator is implicitly assumed in the Crypto-Officer role; otherwise the operator is by default assumed in the User role.

The module does not allow concurrent operators.

4.2. Services

Services are accessed through documented API interfaces from the calling application. The following services listed in Table 6 can be used in the FIPS-approved mode.

Service	Roles		Access to CSPs	API function
	User	Crypto-Officer		
Crypto module initialization/end	✕	✓	No CSP to be accessed	FIPSModuleContext.*cm_init*() FIPSModuleContext.*cm_end*()
Get module status	✓	✓	No CSP to be accessed	FIPSModuleContext.*getState*()
Self-test	✓	✓	Read access to HMAC key.	FIPSModuleContext.*selfTests*()
AES encryption and decryption	✓	✓	Read/write access to the AES symmetric key	FIPSModuleContext.*fips_AES_encrypt*() FIPSModuleContext.*fips_AES_decrypt*()
Triple-DES encryption and decryption	✓	✓	Read/write access to the Triple-DES symmetric key	FIPSModuleContext.*fips_TDES_encrypt*() FIPSModuleContext.*fips_TDES_decrypt*()
DSA domain parameter generation	✓	✓	Write access to the DSA domain parameters	FIPSModuleContext.*fips_DSA2_genPQG*()
DSA domain parameter validation	✓	✓	Read access to the DSA domain parameters	FIPSModuleContext.*fips_DSA2_verPQG*()
DSA Key pair generation	✓	✓	Write access to the DSA key pair	FIPSModuleContext.*fips_DSA2_genKeyPair*()

Service	Roles		Access to CSPs	API function
	User	Crypto-Officer		
DSA Signature generation	✓	✓	Read access to DSA private key	FIPSModuleContext.*fips_DSA2_genSignature*()
DSA Signature verification	✓	✓	Read access to DSA public key	FIPSModuleContext.*fips_DSA2_verSignature*()
RSA key generation	✓	✓	Write access to the RSA key pair	FIPSModuleContext.*fips_RSA_genKeyPair*()
RSA Signature Generation based on PSS (probabilistic signature scheme)	✓	✓	Read access to RSA private key	FIPSModuleContext.*fips_RSA_genSignature*()
RSA Signature Verification based on PSS (probabilistic signature scheme)	✓	✓	Read access to RSA public key	FIPSModuleContext.*fips_RSA_verifySignature*()
SHA-224 SHA-256 SHA-384 SHA-512	✓	✓	No CSP to be accessed	FIPSModuleContext.*fips_SHA2_digest*()
HMAC-SHA224 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512	✓	✓	Read access to HMAC-SHA key	FIPSModuleContext.*fips_HMAC_digest*()
DRBG SP800-90 (Hash_DRBG SHA256)	✓	✓	Write access to seed	FIPSModuleContext. *fips_SHA256DRBG ()*

Table 6: Services that are callable in the FIPS-Approved mode

5. Physical Security

The UEPCM is a software only cryptographic module and thus does not claim any physical security.

6. Operational Environment

6.1. Installation and Invocation

The UEPCM is installed as part of the Unified Element Management Platform on the host system. The validated version of the UEPCM is provided to the ZTE application developers through the uepcm.jar(version 4.11.10).

The module is accessed from its applications by importing the uepcm.jar to its classpath, and it is invoked via the APIs as documented in the UEPCM User's Guide. The following API calling sequence must be conducted in order to initialize the module into the FIPS-Approved mode:

- *FIPSModuleContext*()
- FIPSModuleContext *.cm_init*() to perform the self-tests and turn the module into the FIPS-Approved mode

Figure 3 and Figure 4 below shows the blade system on which UEPCM and its applications run. UEPCM runs on the blade SBCW located at slot 11 and blade SBCO located at slots 1-6 and 9-14.

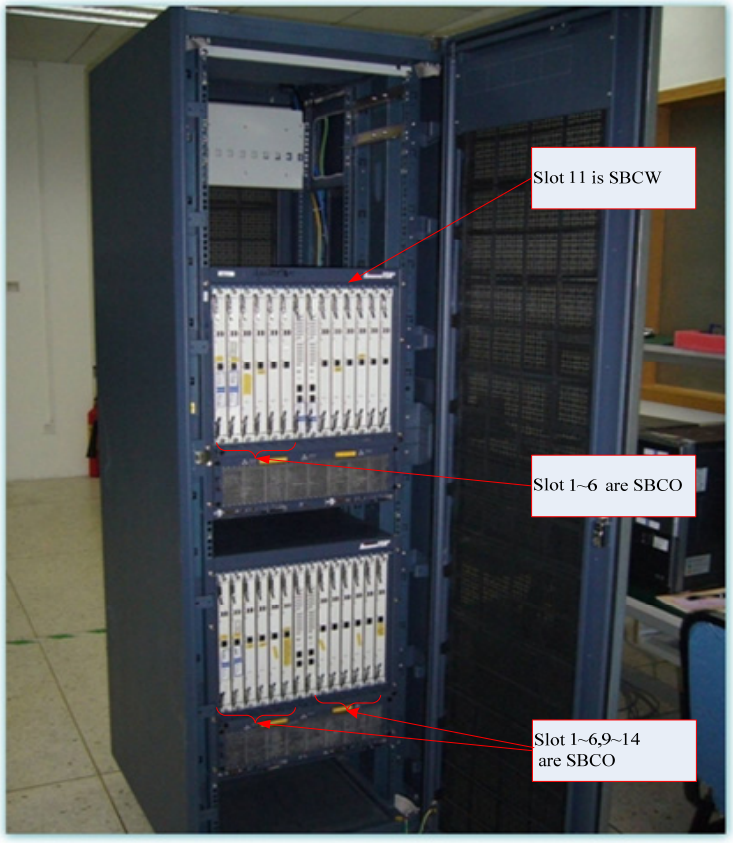

Figure 3: An Overview of a Blade System with the Front Door Open

Figure 4: An Overview of a Blade System with the Back Door Open

6.2. Rules of Operation

The UEPCM will operate in a modifiable operational environment per the FIPS 140-2 definition. The following rules must be adhered to for operating the cryptographic module in a FIPS 140-2 compliant manner:

1. The Operating System shall be restricted to a single human operator mode of operation. The concurrent operators shall be explicitly excluded.
2. The Operating System authentication mechanism must be enabled in order to prevent unauthorized users from being able to access system services.
3. The module must run on a host with commercial grade enclosure and be physically protected as prudent in an enterprise environment.
4. All host system components that can contain sensitive cryptographic data (main memory, system bus, disk storage) must be located within a secure environment.
5. The unauthorized replacement or modification of the legitimate cryptographic module code is not allowed.

6. The application using the module services must utilize a separate copy of the module.
7. The address space of the module must be accessed only by a single process.
8. The unauthorized reading, writing or modification of the application address space which contains the module instance is not allowed.
9. All keys entered into the module must be verified as being legitimate and belonging to the correct entity by software running on the same machines as the module (e.g. the application of the module).
10. The application using the module must access to secret keys, private keys, or other sensitive material through the documented API. Bypassing the API to access any code or data belonging to the cryptographic module is forbidden.

The above rules must be upheld at all times in order to ensure continued system security and FIPS 140-2 compliance after initial setup of the validated configuration. If the module is removed from the above environment, it is assumed not to be operational in the validated mode until such time as it has been returned to the above environment and re-initialized by the Crypto-Officer to the validated condition.

It is the responsibility of the Crypto-Officer to configure the Operating System to operate securely and ensure that only a single operator may operate the module at any particular moment in time.

7. Cryptographic Key Management

7.1. Random Number Generator

The UEPCM implements a FIPS-Approved SP800-90 based Deterministic Random Bit Generator (DRBG) (i.e. Hash_DRBG SHA-256). During initialization, the module obtains a seed and feeds the DRBG (using java.security.SecureRandom to generate the seed randomly, and the DRBG will be reseeded automatically after 1000000 times calls). The UEPCM ensures that the seed key is not identical to the seed, generating the seed key from the seed value hashed with SHA-256. UEPCM performs the continuous test to guarantee that every output is not identical to the previous one.

7.2. Key Generation

The UEPCM uses an FIPS-Approved DRBG SP800-90 as inputs to create the following keys/CSPs:

- RSA keys
- DSA parameters and keys

In the FIPS-Approved mode, the RSA/DSA parameters and key pairs are generated by FIPS-Approved RSA/DSA key generation algorithms. The UEPCM implements the RSA key generation in accordance with the algorithm described in ANSI X9.31, and DSA key generation in accordance with the algorithms described in FIPS 186-3.

7.3. Key Entry and Output

The UEPCM module does not support manual key entry or intermediate key output. In addition, the module does not output keys/CSPs in plaintext format outside its physical boundary.

7.4. Key Storage

The UEPCM, as a software library, does not provide any long-key storage. Keys used by the module are never stored on the hard disk.

7.5. Key Zeroization

Key zeroization is performed via the following API calls:

- Arrays.zeroization():zeroize the byte array.
- DSA2Param.zeroization(): zeroize dsa pqg parameter.

- SecureBigInteger.zeroization(): zeroize DSA2 public key and private key.
- RSAPrivateCrtKeyParameters.zeroization(): zeroize RSA private key.
- RSAKeyParameters.zeroization(): zeroize RSA public key.
- KeyParameter.zeroization(): zeroize AES, TDES, HMAC key.

8. EMI/EMC

The Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC) properties of the UEPCM are not meaningful for the software library itself. Systems utilizing the UEPCM services have their overall EMI/EMC ratings determined by the host hardware platform. The tested platforms for FIPS 140-2 validation have FCC Class A ratings.

9. Self Tests

The UEPCM module implements a number of self-tests to check proper functioning of the module. This includes power-up self-tests (which are also callable on demand) and conditional self-tests.

The self-test can be initiated by calling the API function **FIPSModuleContext.*selfTests*()**. If the self-test is successful, then the module enters the FIPS-Approved operational state. Otherwise, the module enters an error state and returns an error code with applicable description of the error. Once the module is in the error state, API functions that perform cryptographic operations are not available (see Appendix A for the functions available in the error state) and no data output is possible from the module.
When the module is performing self-tests, no API functions are available and no data output is possible until the self-tests are successfully completed.

9.1. Power-Up Tests

The UEPCM module performs self-tests automatically when the API function **FIPSModuleContext.*cm_init*()** is called or on demand when the **FIPSModuleContext.*selfTests*()** is called.

Whenever the power-up tests are initiated, the module performs the integrity test and the cryptographic algorithm Known Answer Test (KAT). If any of these tests fails, the module enters the error state.

9.1.1. Integrity Test

The UEPCM uses FIPS-Approved algorithm HMAC based on SHA-256 for its integrity test. When UEPCM is built, a group of HMAC values are generated for each java class and saved in a fingerprint file mac, which is delivered with uepcm.jar to the UEPCM application developers.
When UEPCM is initialized by its application, it calculates the HMAC value for each class and compares this newly generated HMAC value with the previously saved value. If all pairwise match, then the integrity test is successful. Otherwise, the UEPCM integrity test fails and no services and data outputs from UEPCM shall be available.

9.1.2. Cryptographic algorithm KAT

Upon power-up, a KAT is performed for the following FIPS-Approved algorithms:

- AES cbc, ecb, cfb8, cfb 128, ofb mode encryption/decryption;
- Triple-DES cbc, cfb8, cfb64, ecb, ofb mode encryption/decryption;
- RSA 1536 bits(SHA-224, SHA-256) PSS signature generation/verification
- DSA (L=2048, N=224, SHA-224, SHA-256)signature generation/verification
- SHA256_DRBG

- SHA-224, SHA-256, SHA-384, SHA-512
- HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512

9.2. Conditional Tests

9.2.1. Pair-wise consistency test

The UEPCM module performs the pair-wise consistency test for each pair of RSA/DSA keys that it generates. The consistency of the key pair is tested by the calculation and verification of a digital signature. If the digital signature cannot be verified, the test fails.

9.2.2. Continuous test for DRBG

The UEPCM module implements a continuous test for the DRBG SP800-90. The DRBG generates a string of random bits per request. The generated random bits is compared with the random value generated for the previous request. If the generated data for two requests are identical, a conditional test error flag is raised.

10. Design Assurance

10.1. Configuration Management

The UEPCM module develop team utilizes Subversion (SVN), a software versioning and a revision control system, to maintain current and historical versions of files such as source code and design documentation that contribution to the UEPCM.

SVN integrates several aspects of the software development process in a distributed development environment to facilitate project-wide coordination of development activities across all phases of the product development life cycle:

- Configuration Management – the process of identifying, managing and controlling software modules as they change over time.
- Version Control – the storage of multiple versions of a single file along with information about each version.
- Change control – centralizes the storage of files and controls changes to files through the process of checking files in and out.

The list of files that are relevant to the UEPCM module and subject to SVN control is detailed in the ZTE UEPCM component list document.

10.2. Guidance

10.2.1. Cryptographic Officer Guidance

A Crypto officer uses the installation instructions provided in the ZTE UEPCM User's Guide to install the module in their environment.

To bring the module into FIPS-Approved mode, the crypto officer has to carry out the startup steps as specified in the ZTE UEPCM User's Guide and follow the rules of operations as stated in Section 6 of this Security Policy.

10.2.2. User Guidance

The ZTE application developers using the UEPCM services must verify that ownership of keys is not compromised, and keys are not shared between different users of the calling application. Note that this requirement is not enforced by the UEPCM itself. It is the responsibility of the application to provide the verified keys to the UEPCM.

The ZTE application developers are referred to the ZTE UEPCM User's Guide for details in regard to integrating UEPCM module into their applications.

11. Mitigation of Other Attacks

No other attacks are mitigated.

12. Appendix A

The module API functions are fully described in the ZTE UEPCM User's Guide. The following is the list of the module API functions serving as a quick reference for the completion of this document.

Y - Functions that can be performed in FIPS mode

N - Functions that cannot be performed in FIPS mode.

E - Functions that can be performed in error state.

FIPSModuleContext.FIPSModuleContext()	N
FIPSModuleContext.cm_init()	N
FIPSModuleContext.cm_end()	YE
FIPSModuleContext.selfTests()	YE
FIPSModuleContext.getState()	YE
IntegrityTest.test()	YE
FIPSModuleContext.fips_AES_encrypt()	Y
FIPSModuleContext.fips_AES_decrypt()	Y
FIPSModuleContext.fips_TDES_encrypt()	Y
FIPSModuleContext.fips_TDES_decrypt()	Y
FIPSModuleContext.fips_SHA2_digest()	Y
FIPSModuleContext.fips_HMAC_digest()	Y
FIPSModuleContext.fips_SHA256DRBG()	Y
FIPSModuleContext.fips_RSA_genKeyPair()	Y
FIPSModuleContext.fips_RSA_genSignature()	Y
FIPSModuleContext.fips_RSA_verifySignature()	Y
FIPSModuleContext.fips_DSA2_GenPQG()	Y
FIPSModuleContext.fips_DSA2_verPQG()	Y
FIPSModuleContext.fips_DSA2_genKeyPair()	Y
FIPSModuleContext.fips_DSA2_genSignature()	Y
FIPSModuleContext.fips_DSA2_verSignature()	Y
Arrays.zeroization()	YE
SecureBigInteger.zeroization()	YE
DSA2Param.zeroization()	YE
RSAPrivateCrtKeyParameters.zeroization()	YE
RSAKeyParameters.zeroization()	YE
KeyParameter.zeroization()	YE