ZTE Unified Platform Cryptographic Library Version 1.1

FIPS 140-2 Non-Proprietary Security Policy

Policy Version 1.9

Last Updated: 2011-07-09

Revision history

Version	Change date	Author(s)	Changes to previous version
0.1	2010-10-25	atsec	First skeleton draft.
0.2	2010-11-10	atsec	First draft for ZTE to fill in missing information.
0.3	2010-12-03	ZTE upcl team	Filled in the missing information and clarified some questions.
0.4	2010-12-04	atsec	Made a revision based on the ZTE inputs provided in v0.3.
0.5	2010-12-14	ZTE upcl team	Add some information
0.6	2011-01-04	atsec	Made a revision based on the ZTE inputs provided in v0.5.
0.7	2011-1-13	ZTE upcl team	Add figures of testing environment,update the version of UPCL to 1.1
0.8	2011-01-18	atsec	Updates to reflect the NIST algorithm transition requirements. Updated section 6 to include blade system pictures that are identical to those used in the FCC report.
0.9	2011-01-18	ZTE upcl team	Add some information for the NIST algorithm transition
1.0	2011-01-28	ZTE upcl team	Modify some information
1.1	2011-02-19	ZTE upcl team	Modify some information
1.2	2011-02-24	ZTE upcl team	Modify some information
1.3	2011-03-01	ZTE upcl team	Modify some information
1.4	2011-03-02	ZTE upcl team	Modify some information
1.5	2011-03-04	atsec	A final walk through of SP
1.6	2011-03-07	atsec	Removed non-FIPS approved algorithms that are ONLY available in the non-FIPS mode.
1.7	2011-06-28	atsec	Modify usage of EC_KEY_free() function in Appendix A.
1.8	2011-07-01	atsec	Updated figure 2 by removing the OpenSSL box. Add non-FIPS approved algorithms that are only available in the non-FIPS mode.
1.9	2011-07-09	atsec	Included ECDH in section 7.3 Key Establishment and also indicated that ECDH is only available in the non-FIPS mode.

Table of contents

1 Introduction 5
2 Cryptographic Module Specification 6
2.1 Module Overview 6
2.2 Cryptographic Module Description 7
2.3 Cryptographic Module Security Level 9
2.4 Tested Platforms 9
2.5 Approved Mode(s) of operation 10
3 Cryptographic Module Ports and Interfaces 12
4 Roles, Services and Authentication 13
4.1 Roles 13
4.2 Services 13
5 Physical Security 17
6 Operational Environment 18
6.1 Installation and Invocation 18
6.2 Rules of Operation 19
7 Cryptographic Key Management 21
7.1 Random Number Generator 21
7.2 Key Generation 21
7.3 Key Establishment 21
7.4 Key Entry and Output 21
7.5 Key Storage 21
7.6 Key Zeroization 21
8 EMI/EMC 23
9 Self-Tests 24
9.1 Power-Up Tests 24
9.1.1 Integrity Test 24
9.1.2 Cryptographic algorithm KAT 24
9.2 Conditional Tests 24
9.2.1 Pair-wise consistency test 24
9.2.2 Continuous test for DRGB 25
10 Design Assurance 26
10.1 Configuration Management 26
10.2 Guidance 26
10.2.1 Cryptographic Officer Guidance 26
10.2.2 User Guidance 26

11 Mitigation of Other Attacks 27
Appendix A 28

1 Introduction

This document is a non-proprietary FIPS 140-2 Security Policy for the Unified Platform Cryptographic Library ("UPCL" or "module") version 1.1 cryptographic module. 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 UPCL White Book and UPCL API USER GUIDE are technical references for the UPCL application developers to use and integrate the UPCL 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 Crypto 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 UPCL 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 Platform Cryptographic Library is a software only cryptographic module that is dynamically linked into several telecommunication applications developed by ZTE. The UPCL provides cryptographic services to applications in supporting their implementation of network protocols like IPSEC, IKE and TLS so that these applications can achieve the peer authentication and data protection as needed. A typical deployment environment of the UPCL and its applications are shown in Figure 1 below.

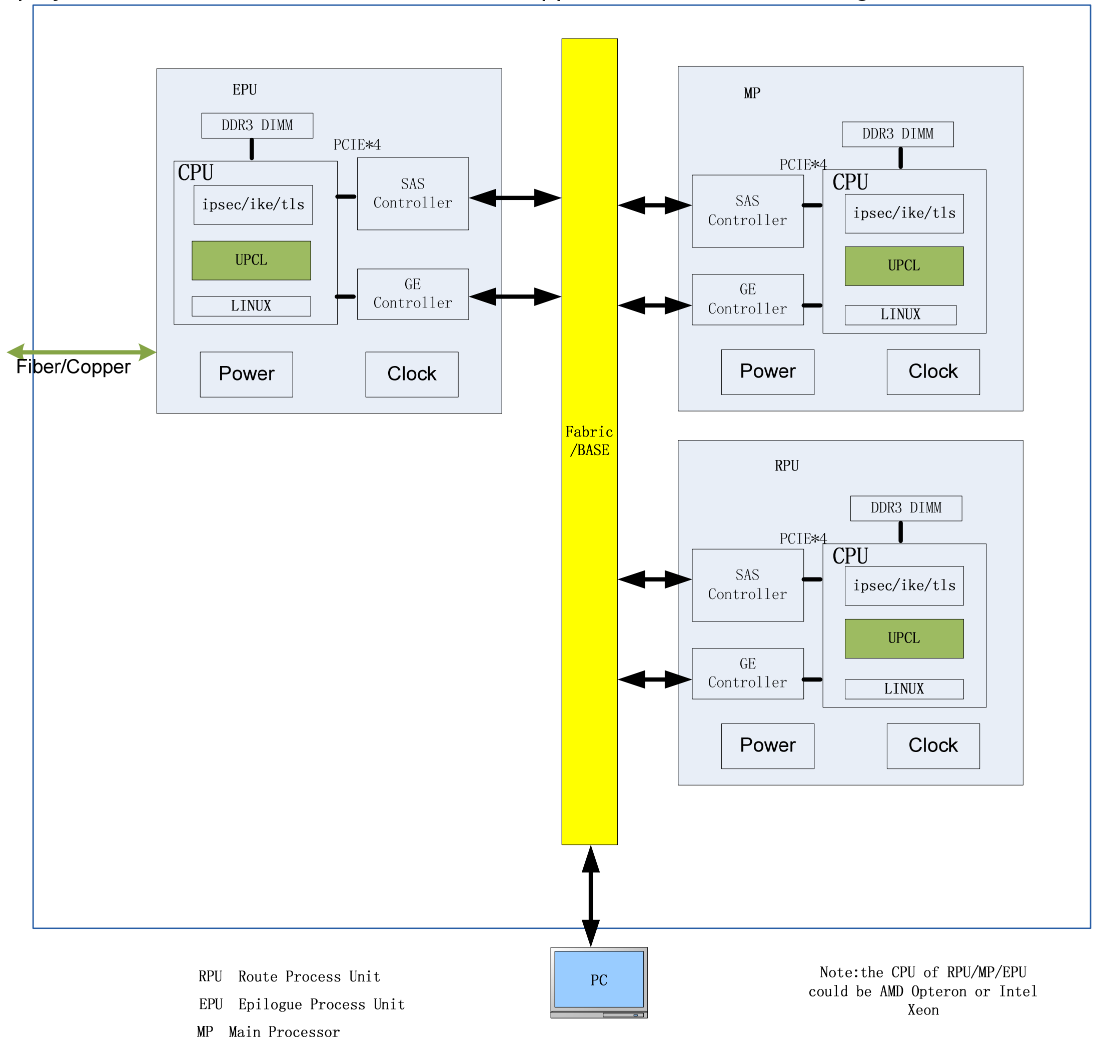

Figure 1: A Typical Deployment Environment of the UPCL and Its ZTE Applications

The module and its applications are deployed on a blade system as part of the telecommunication infrastructure such as ZXUR 9000 C,ZXUR 9000 TD,ZXUR 9000 UMTS,ZXUR 9000 GSM BSC ,ZXWN HLR,ZXWN MSCS,ZXC10 MSCe,ZXUN Mss,ZXUN MGW,ZXUN iLIG,ZXC10 HLRe,ZXC10 FLR,ZXC10 OTAFe,ZXUN HSS,ZXUN UniA,ZXUN USPP, ZXUN iSTP,ZXUN SSS,ZXUN RCP,ZXUN CG,ZXUN eCG,ZXUN CSCF,ZXUN B200,ZXUN AGCF,ZXSS10 SS1b,ZXSS10 B200 etc that carries the land or wireless telecommunication networks. The Operating System on which the module and its applications run is EMBSYS™ Carrier Grade Embedded Linux V3 with Linux kernel version 2.6.21.

EPU, RPU and MP in Figure 1 represent three different types of hardware board that may be included in a Blade system. On each type of the above mentioned three hardware boards, the CPU is either Intel® Xeon™ or AMD Opteron®.

2.2 Cryptographic Module Description

The physical boundary of the UPCL cryptographic module is defined as the opaque enclosure surrounding 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. RPU, EPU and MP in Figure 1 are three models of blades that serve as hardware platforms for the UPCL module and its application.

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

Component Type	File Name	Version
Dynamic Library binary code	lib_upcl.so	1.1
Header File	upcl_pub.h	1.1
Documentation	ZTE UPCL White Book	1.10
	UPCL API USER GUIDE	1.7

Table 1: The UPCL Cryptographic Module Components

The UPCL cryptographic module is developed in C language and is provided to application developers as a dynamically linkable library with a C header file. Applications interact with the cryptographic module through an API (see appendix A).

The relationship between the UPCL and its ZTE applications is shown in the following diagram. The UPCL comprises a C header file and a dynamically linkable library binary code.

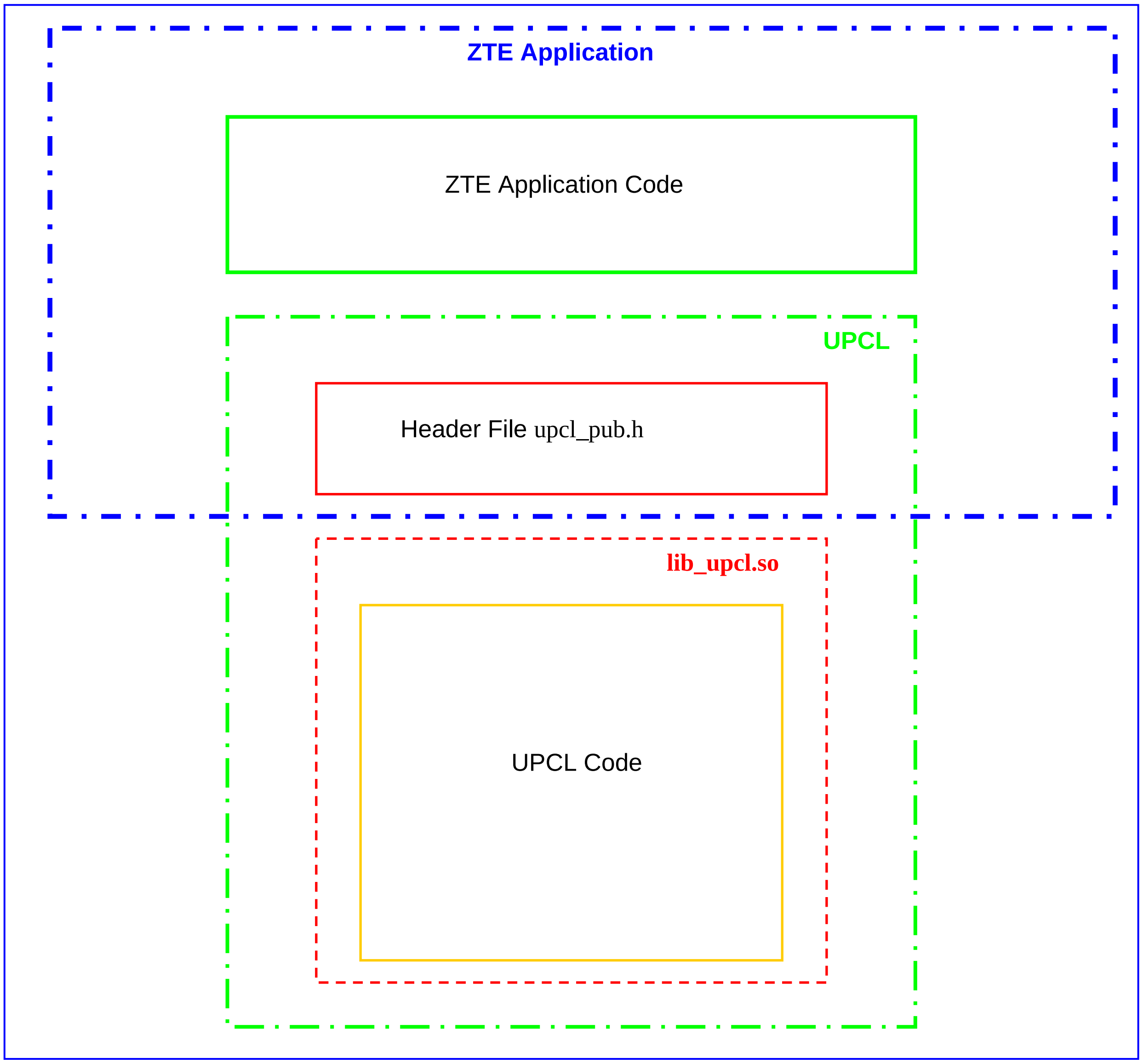

Figure 2: Functional Decomposition of the UPCL and Its ZTE Applications

- ZTE Application – The ZTE application using the UPCL. The ZTE application is built upon the application code and the UPCL header file which dynamically links to UPCL binary code.
- ZTE Application code – The program using the UPCL to perform cryptographic functions.
- UPCL – The cryptographic module to be validated that contains C header files and a dynamic library lib_upcl.so that can be linked to the ZTE application.
- lib_upcl.so – a dynamically linkable cryptographic library that is compiled from the UPCL code, which is largely based on the OpenSSL code.

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 3 - Security Levels for Eleven Sections of the FIPS 140-2 Requirements

2.4 Tested Platforms

The UPCL validation was performed on the following platforms.

Manufacturer	Model	CPU	EMI/EMC Compliance	OS and Version
ZTE	SBCO	AMD Opteron® 64	FCC report provided by ZTE CORPORATION EMC Laboratory	EMBSYS™ Carrier Grade Embedded Linux V3
ZTE	SBCW	Intel® Xeon™ 64	FCC report provided by ZTE CORPORATION EMC Laboratory	EMBSYS™ Carrier Grade Embedded Linux V3

Table 4 - Tested platforms

2.5 Approved Mode(s) of operation

The UPCL module implements a list of FIPS-Approved algorithms or callable in FIPS-mode as shown in Table 5.

Algorithm w/modes	Keys/CSPs	Usage	Algorithm Certificate #
AES (ECB, CBC, OFB, CFB8, CFB128)	128, 192, 256 bit keys	Encryption/decryption	INTEL:#1586 AMD: #1585
Triple-DES (ECB, CBC, CFB, OFB)	3-key 168 bits	Encryption/decryption	INTEL:#1042 AMD: #1041
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:#492 AMD: #491
RSA (ANSI X9.31 (Key(gen), SIG(gen), SIG(ver)); PKCS #1.5 (SIG(gen), SIG(ver)); PSS (SIG(gen), SIG(ver)))	Module sizes: 1536, 2048, 3072, 4096; Public Key sizes: 65537	Key Pair generation, Generate and Verify signature	INTEL:#776 AMD: #775
DRBG SP800-90 (Hash_DRBG SHA256)	seed value and 256 bit seed key value.	Random bit generation.	INTEL:#76 AMD: #75
SHA-224 (BYTE-only) SHA-256 (BYTE-only) SHA-384 (BYTE-only) SHA-512 (BYTE-only)	N/A	Hashing	INTEL:#1405 AMD: #1404
HMAC-SHA-224 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512	HMAC keys Key Size Ranges Tested: Key Size < Block Size, Key Size = Block Size, Key Size > Block Size	Message Authentication	INTEL:#932 AMD: #931
Diffie-Hellman	1024-4096 bits key	Key establishment	None

Table 5: FIPS-Approved/Allowed Cryptographic Algorithms

The UPCL module also implements a list of non-FIPS-Approved algorithms and only used in non-FIPS mode as shown in the table below.

Algorithm	Keys/CSPs	Usage
EC/ECDH	P: 192-521 bits K and B: 163-571 bits	Key establishment
IDEA	IDEA keys	Encrypt and Decrypt
Blowfish	Blowfish keys	Encrypt and Decrypt
DES	DES 56 bit keys	Encrypt and Decrypt

RC2	N/A	Hashing
RC4	N/A	Hashing
MD2	HMAC keys	Message Authentication
MD4	HMAC keys	Message Authentication
MD5	HMAC keys	Message Authentication
Ripemd	HMAC keys	Message Authentication
CAST	CAST keys	Encrypt and Decrypt

Table 6: non-FIPS-Approved Cryptographic Algorithms

The module can be operated both in FIPS mode and non-FIPS mode. In FIPS mode, the module provides the FIPS-Approved algorithms and FIPS-Allowed Diffie-Hellman as listed in Table 5. Any invocation of the non-FIPS-Approved algorithms as shown in Table 6 will cause the module enter the non-FIPS mode. Algorithms listed in Table 6 are not available in the FIPS-mode.

The module maintains a process flag to indicate that the module is in FIPS mode. The flag is provided in g_upcl_state. If the flag contains a value of 2, the module operates in FIPS mode. The user can query at any time about the mode in which the module operates by invoking the UPCL_get_state() function.

3 Cryptographic Module Ports and Interfaces

As a software only cryptographic module, the logical interface of the UPCL is its API documented in the ZTE UPCL White Book and UPCL API USER GUIDE. The execution of the UPCL module 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 UPCL logical interfaces originated in its API.

FIPS Required Interface	The UPCL 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 7: Ports and Interface of the UPCL

The UPCL 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 UPCL does not support cryptographic bypass capability, either.

Any internal error detected by the UPCL 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 upcl_pub.h. They do not reveal any sensitive material to callers.

4 Roles, Services and Authentication

4.1 Roles

The UPCL 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 services listed in the following table can be used in the FIPS mode with proper parameters to invoke the FIPS-approved algorithms.

Service	Roles		Access to CSPs	API function
	User	Crypto-Officer		
Crypto module initialization/end	No	Yes	No CSP to be accessed	UPCL_init UPCL_end
get module status	Yes	Yes	No CSP to be accessed	UPCL_get_state UPCL_get_errcode
Self-test	Yes	Yes	Read access to HMAC key.	UPCL_fips_selftest
AES encryption and decryption	Yes	Yes	Read/write access to the AES symmetric key	UPCL_sectx_init UPCL_sectx_set UPCL_sectx_update UPCL_sectx_final UPCL_sectx_clear UPCL_sectx_encrypt UPCL_encrypt

Service	Roles		Access to CSPs	API function
	User	Crypto-Officer		
Triple-DES encryption and decryption	Yes	Yes	Read/write access to the Triple-DES symmetric key	UPCL_sectx_init UPCL_sectx_set UPCL_sectx_update UPCL_sectx_final UPCL_sectx_clear UPCL_sectx_encrypt UPCL_encrypt
DSA domain parameter generation	Yes	Yes	Write access to the DSA domain parameters	UPCL_dsa_generate_parameters_ex
DSA domain parameter validation	Yes	Yes	Read access to the DSA domain parameters	UPCL_dsa_verify_parameters_ex
DSA Key pair generation	Yes	Yes	Write access to the DSA key pair	UPCL_dsa_generate_key FIPS_dsa_new FIPS_dsa_free UPCL_PKEY_size
DSA Signature generation	Yes	Yes	Read access to DSA private key	UPCL_hactx_init UPCL_sign_set UPCL_sign_update UPCL_sign_final UPCL_sign UPCL_hactx_clear
DSA Signature verification	Yes	Yes	Read access to DSA public key	UPCL_hactx_init UPCL_verify_set UPCL_verify_update UPCL_verify_final UPCL_verify UPCL_hactx_clear
RSA key generation	Yes	Yes	Write access to the RSA key pair	UPCL_rsa_x931_derive_ex UPCL_rsa_x931_generate_key_ex UPCL_rsa_generate_key_ex FIPS_rsa_new FIPS_rsa_free UPCL_PKEY_size

Service	Roles		Access to CSPs	API function
	User	Crypto-Officer		
RSA signature Generation based on X9.31	Yes	Yes	Read access to RSA private key	UPCL_hactx_init UPCL_sign_set UPCL_sign_update UPCL_sign_final UPCL_sign UPCL_hactx_clear
RSA Signature Verification based on X9.31	Yes	Yes	Read access to RSA public key	UPCL_hactx_init UPCL_verify_set UPCL_verify_update UPCL_verify_final UPCL_verify UPCL_hactx_clear
SHA-224 SHA-256 SHA-384 SHA-512	Yes	Yes	No CSP to be accessed	UPCL_hactx_init UPCL_hactx_set UPCL_hactx_update UPCL_hactx_final UPCL_hactx_clear UPCL_hash
HMAC-SHA224 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512	Yes	Yes	Write access to HMAC-SHA key	UPCL_hmctx_init UPCL_hmctx_set UPCL_hmctx_update UPCL_hmctx_final UPCL_hmctx_clear UPCL_hmctx_hmac
DRBG SP800-90 (Hash_DRBG SHA256）	Yes	Yes	Write access to seed	UPCL_fips_rand_bytes UPCL_fips_rand_status UPCL_rand_bytes UPCL_rand_add UPCL_rand_seed UPCL_rand_pseudo_bytes UPCL_rand_status UPCL_rand_cleanup

Service	Roles		Access to CSPs	API function
	User	Crypto-Officer		
Diffie-Hellman (DH) parameter generation, key generation, key agreement	Yes	Yes	Read/write access to DH public and private key	UPCL_dh_generate_parameters_ex UPCL_dh_generate_key UPCL_dh_compute_key FIPS_dh_new FIPS_dh_free

Table 8: Services that are callable in the FIPS-Approved mode

5 Physical Security

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

6 Operational Environment

6.1 Installation and Invocation

The UPCL is installed as part of the ZTE applications that implement network protocols such as IPSEC, IKE and TLS on the host system. The validated version of the UPCL is provided to the ZTE application developers through the delivery package include lib_upcl.so and lib_upcl.so.sha256.

The module is accessed from its applications through the inclusion of header file upcl_pub.h, and it is invoked via the APIs as documented in the ZTE UPCL White Book and UPCL API USER GUIDE.

To initialize the UPCL, its application must call UPCL_init(1) 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 UPCL and its applications run. UPCL runs on the blade SBCW located at slot 11 and blade SBCO located at slots 1-6 and 9-14.

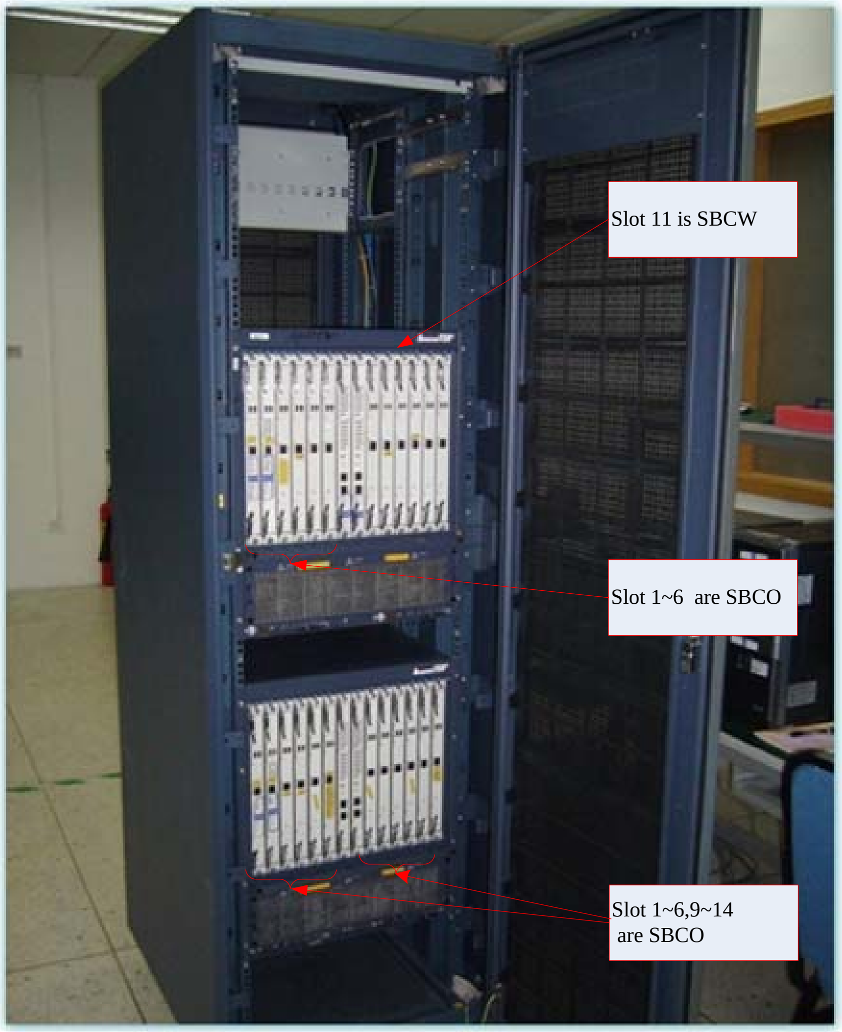

Figure 3: An Overview of a Blade System with the Front Door Open

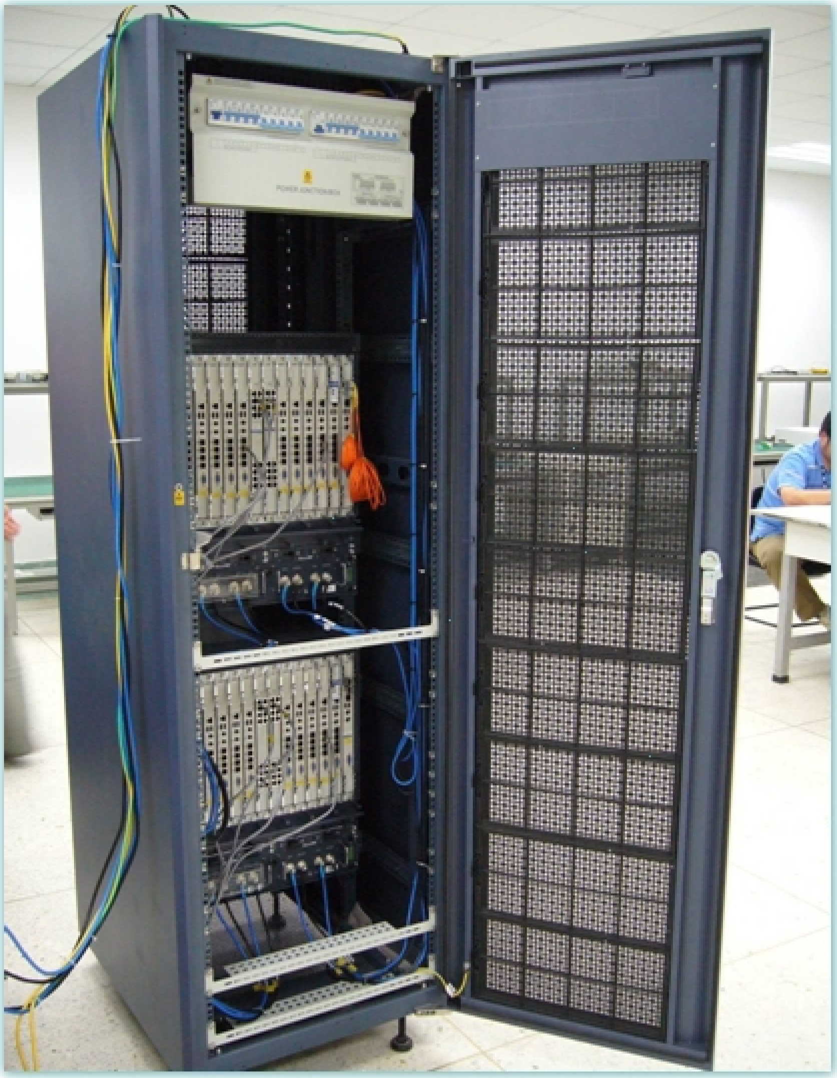

Figure 4: An Overview of a Blade System with the Back Door Open

6.2 Rules of Operation

The UPCL 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 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 UPCL implements a FIPS-Approved SP800-90 based Deterministic Random Bit Generator (DRBG) (i.e. Hash_DRBG SHA-256). During initialization, the crypto module obtains a seed and feeds the DRBG (the calling application may reseed the DRBG at any time by providing a seed as an input parameter). The UPCL module ensures that the seed key is not identical to the seed, generating the seed key from the seed value hashed with SHA-256. UPCL performs the continuous test to guarantee that every 32-byte output is not identical to the previous one.

7.2 Key Generation

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

- RSA keys
- DSA parameters and keys
- DH 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 UPCL 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.

UPCL may also use a DRBG SP800-90 output for AES or Triple-DES key, if the calling application chooses to do that.

7.3 Key Establishment

The UPCL provides its application the following key establishment methodologies:

- Diffie-Hellman (DH) with 1024-4096 bits key to provide 80-150 bits of security strength
- Elliptic Curve Diffie-Hellman (ECDH) with 163-571 bits curve to provide 80-256 bits of security strength

DH can be used in the FIPS-mode, while ECDH is only available in the non-FIPS mode. In order to be FIPS 140-2 compliant, it is the calling application’s responsibility to use DH with a right key size that has security strength greater than or equal to the security strength of the key to be established. The guidance for the comparable security strengths can be found in Table 2 in NIST SP800-57.

7.4 Key Entry and Output

The UPCL 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.

7.5 Key Storage

The UPCL module, as a software library, does not provide any long-key storage. Keys used by the module are ever stored on the hard disk.

7.6 Key Zeroization

The UPCL module modifies the default OpenSSL scrubbing code to zero objects instead of filling with pseudo random data and adds explicit testing for zeroization. Key zeroization is performed via the following API calls:

- FIPS_dsa_free: zeroize and free dsa key
- FIPS_rsa_free: zeroize and free rsa key
- UPCL_sectx_clear: zeroize and free symmetric key context
- UPCL_hactx_clear: zeroize and free hash context
- UPCL_hmctx_clear: zeroize and free hmac context
- BN_clear_free: zeroize and free BN(big number)
- FIPS_dh_free: zeroize and free dh key
- EC_KEY_free: zeroize and free ec key

8 EMI/EMC

The Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC) properties of the UPCL are not meaningful for the software library itself. Systems utilizing the UPCL 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 UPCL 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 UPCL_fips_selftest(). If the self-test is successful, then the module enters the "FIPS" state, which is the FIPS-approved operational state. Otherwise, the module enters the "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 UPCL module performs self-tests automatically when the API function UPCL_init(1) is called or on demand when the API function UPCL_fips_selftest() 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 UPCL uses FIPS-Approved algorithm HMAC based on SHA-256 for its integrity test. When UPCL is built, an HMAC value is generated and saved in a fingerprint file lib_upcl.so.sha256, which is delivered with lib_upcl.so to the UPCL application developers.

When UPCL is initialized by its application, it calculates the HMAC value again and compares this newly generated HMAC value with the previously saved value. If two values are the same, then the integrity test is successful. Otherwise, the UPCL integrity test fails and no services and data outputs from UPCL shall be available.

9.1.2 Cryptographic algorithm KAT

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

- AES 128 ECB encryption/decryption
- 3-key Triple-DES ECB encryption/decryption
- RSA 2048 bits (SHA-224, SHA-256, SHA-384, SHA-512) PKCS#1, (SHA-224, SHA-256, SHA-384, SHA-512) PSS, (SHA-224, SHA-256, SHA-384, SHA-512) X9.31 signature generation/verification
- DSA (2048 bits, N = 256, SHA-512) SHA-256 signature generation/verification
- DRBG SP800-90 SHA-256
- HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 (SHA-224, SHA-256, SHA-384, SHA-512 KAT are embedded into HMAC)

9.2 Conditional Tests

9.2.1 Pair-wise consistency test

The UPCL 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 DRGB

The UPCL module implements a continuous test for the DRBG SP800-90. The DRBG generates a minimum of 32 bytes per request. The 55 bytes internal state data generated for every request is compared with the 55 bytes data generated for the previous request. If the generated data for two requests are identical, a conditional test error flag is raised. For the first request made to any instantiation of the DRNG implemented in the module, two sets of 55 bytes internal state data are generated and then compared. A conditional test error flag is raised if they are identical.

10 Design Assurance

10.1 Configuration Management

The UPCL 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 UPCL.

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 UPCL module and subject to SVN control is detailed in the ZTE UPCL White Book.

10.2 Guidance

10.2.1 Cryptographic Officer Guidance

A Crypto officer uses the installation instructions provided in the ZTE UPCL White Book to install the module in their environment. The integrity of the UPCL is automatically verified during the installation. If the Crypto officer observes an integrity error, then s/he shall not install the module.

To bring the module into FIPS-Approved mode, the crypto officer has to carry out the startup steps as specified in the ZTE UPCL White Book 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 UPCL 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 UPCL itself. It is the responsibility of the application to provide the verified keys to the UPCL.

The ZTE application developers using the UPCL services must avoid using non-FIPS-Approved algorithms or non-FIPS-Approved mode of operation. If not feasible, the applications must indicate that they utilize non-FIPS-Approved cryptographic services via the FIPS mode indicator.

The ZTE application developers are referred to the ZTE UPCL White Book for details in regard to integrating UPCL module into their applications.

11 Mitigation of Other Attacks

No other attacks are mitigated.

Appendix A

The module API functions are fully described in the ZTE UPCL White Book. 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.

UPCL_init	N	BN_new	YE
UPCL_end	YE	BN_bin2bn	YE
UPCL_fips_selftest	YE	BN_free	YE
UPCL_get_state	YE	BN_clear_free	YE
UPCL_get_errcode	YE	FIPS_rsa_new	Y
UPCL_malloc	YE	FIPS_rsa_free	YE
UPCL_free	YE	UPCL_rsa_x931_derive_ex	Y
UPCL_sectx_init	Y	UPCL_rsa_x931_generate_key_ex	Y
UPCL_sectx_set	Y	UPCL_rsa_generate_key_ex	Y
UPCL_sectx_update	Y	UPCL_PKEY_size	Y
UPCL_sectx_final	Y	UPCL_sign_set	Y
UPCL_sectx_encrypt	Y	UPCL_sign_update	Y
UPCL_sectx_clear	Y	UPCL_sign_final	Y
UPCL_encrypt	Y	UPCL_sign	Y
UPCL_hactx_init	Y	UPCL_verify_set	Y
UPCL_hactx_set	Y	UPCL_verify_update	Y
UPCL_hactx_update	Y	UPCL_verify_final	Y
UPCL_hactx_final	Y	UPCL_verify	Y
UPCL_hactx_clear	Y	UPCL_dsa_generate_parameters_ex	Y
UPCL_hash	Y	UPCL_dsa_verify_parameters_ex	Y
UPCL_hmctx_init	Y	UPCL_dsa_generate_key	Y
UPCL_hmctx_set	Y	FIPS_dsa_new	Y
UPCL_hmctx_update	Y	FIPS_dsa_free	YE
UPCL_hmctx_final	Y	UPCL_dh_generate_parameters_ex	Y
UPCL_hmctx_hmac	Y	UPCL_dh_generate_key	Y
UPCL_hmctx_clear	Y	UPCL_dh_compute_key	Y
UPCL_fips_rand_bytes	Y	FIPS_dh_new	Y
UPCL_fips_rand_status	Y	FIPS_dh_free	YE
UPCL_rand_bytes	Y	EC_KEY_new_by_curve_name	N
UPCL_rand_add	Y	EC_KEY_generate_key	N
UPCL_rand_seed	Y	EC_KEY_new	N
UPCL_rand_cleanup	Y	EC_KEY_free	N
UPCL_rand_pseudo_bytes	Y	ECDH_compute_key	N
UPCL_rand_status	Y	EC_KEY_get0_public_key	N