IBM® z/OS® Version 1 Release 13 ICSF PKCS #11 Cryptographic Module

FIPS 140-2

Non-Proprietary Security Policy

Policy Version 2.00

IBM Systems & Technology Group
System z Development
Poughkeepsie, New York

December 12, 2011

Table of Contents

Scope of Document

This document describes the services that the z/OS Integrated Cryptographic Service Facility PKCS #11 module ("ICSF PKCS #11" or "module") provides to security officers and end users, and the policy governing access to those services. It complements official product documentation, which concentrates on application programming interface (API) level usage and environmental setup. [1]

Module Description The z/OS ICSF PKCS #11 module in its FIPS 140-2 configuration consists of a set of loadable modules. The deployed version consists of the following modules:

Table 1 ICSF PKCS #11 Module Modules

Core	Auxiliary			
CSFINPV2	CSFINPVT	CSFPPRF	CSFPHMV	CSFDLL64
IRRPVERS	CSFPDVK	CSFPPKV	CSFPWPK	CSNPCA3X
	CSFPDMK	CSFPSKD	CSNPCAPI	CSNPCI3X
	CSFPHMG	CSFPSKE	CSNPCINT	CSNPCU3X
	CSFPGKP	CSFPSAV	CSNPCUTL	CSFDLL3X
	CSFPGSK	CSFPTRC	CSFDLL	IRRVERLD
	CSFPGAV	CSFPTRD	CSNPCA64	Header Files
	CSFPOWH	CSFPTRL	CSNPCI64	Side Decks
	CSFPPKS	CSFPUWK	CSNPCU64	Sample Application

The z/OS ICSF PKCS #11 install package consists of the core modules that are utilized while operating in FIPS 140-2 mode, as well as some auxiliary modules and files. The auxiliary modules provide functionality that is not cryptographically relevant (i.e., pass-through modules). The files consist of header, side decks, sample application, and sample configuration scripts.

The z/OS ICSF PKCS #11 logical and physical boundaries are described in Figures 1 and 2 in the Operational Environment Section.

Cryptographic Module Specification

The z/OS ICSF PKCS #11 module is classified as a *multi-chip standalone software-hybrid module* for **FIPS Pub 140-2** purposes. The actual cryptographic boundary for this FIPS 140-2 module validation includes the ICSF PKCS #11 module running in configurations backed by hardware cryptography. The ICSF PKCS #11 module consists of software-based cryptographic algorithms, as well as symmetric and hashing algorithms provided by the CP Assist for Cryptographic Function (CPACF) and RSA Hardware clear key modular math cryptography provided through the Crypto Express3 card (CEX3A or 4765-001). The RSA hardware support is accessed through auxiliary module CSFINPVT which acts as a "pipe" between ICSF PKCS #11 and the cryptographic cards.

ICSF PKCS #11 testing was performed using the z/OS Version 1 Release 13 operating system with the following platform configurations:

1. IBM zEnterprise™ 196 (z196) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 (Base GPC)

2. IBM zEnterprise 196 (z196) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and optional Crypto Express3 card (CEX3A)

The module running on the above platforms met all **FIPS Pub 140-2** Level 1 security requirements.

In addition to the configurations tested by the laboratory, vendor-affirmed testing was performed on the following platforms and operating system levels:

- Using z/OS Version 1 Release 13

1. IBM zEnterprise™ 114 (z114) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 (Base GPC)

2. IBM zEnterprise 114 (z114) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and optional Crypto Express3 card (CEX3A)

Security level This document describes the security policy for the z/OS ICSF PKCS #11 with Level 1 overall security as defined in **FIPS Pub 140-2** [2].

Table 2 ICSF PKCS #11 Module Components

Type / Names	**Version**
Software components ICSF CSFINPV2 RACF IRRPVERS	 ICSF level HCR7780 with APAR OA36882 RACF level HRF7780
Hardware components CPACF 4765-001	 Firmware – CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 (aka FC3863) with System Driver Level 86E Hardware – COP chips integrated within processor unit Firmware – level e1ced7a0 Hardware – P/N 45D6048
Documentation	APAR OA34403 PDF (prerequisite APAR for OA36882) *z/OS Cryptographic Services Integrated Cryptographic Service Facility Overview* (SA22-7519-14) *z/OS Cryptographic Services Integrated Cryptographic Service Facility System Programmer's Guide* (SA22-7520-15) *z/OS Cryptographic Services Integrated Cryptographic Service Facility Application Programmer's Guide* (SA22-7522-14) *z/OS Cryptographic Services Integrated Cryptographic Service Facility Administrator's Guide* (SA22-7521-15) *z/OS Cryptographic Services Integrated Cryptographic Service Facility Writing PKCS #11 Applications* (SA23-2231-03)

Cryptographic Module Security Level

The module is intended to meet requirements of Security Level 1 overall, with certain categories of security requirements not applicable (Table 3).

Table 3 Module Security Level Specification

Security Requirements Section	**Level**
Cryptographic Module Specification	3
Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	1
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of other attacks	N/A
Overall	1

Ports and Interfaces

As a multi-chip standalone module, the ICSF PKCS #11 physical interfaces are the boundaries of the host running ICSF PKCS #11 module code. The underlying logical interfaces of the module are the PKCS #11 callable services documented in the OA34403.pdf and the *z/OS Cryptographic Services Integrated Cryptographic Service Facility Application Programmer's Guide* and the RACF callable service documented in the *z/OS Security Server RACF Callable Services* manual.

Control inputs which control the mode of the module are provided through the FIPSMODE ICSF startup option and a vendor defined PKCS #11 key attribute, CKA_IBM_FIPS140.

Data input and data output are provided in the variables passed on callable service invocations, generally through user-supplied buffers. Hereafter, the callable services will be referred to as "API".

Status output is provided in return and reason codes and through messages. Documentation for the API lists the return and reason codes. A complete list of all return and reason codes returned by the APIs within the ICSF PKCS #11 module is provided in the ICSF PKCS #11 reference manual, *z/OS Cryptographic Services Integrated Cryptographic Service Facility Application Programmer's*.

Cryptographic bypass capability is not supported by ICSF PKCS #11.

Module Status The ICSF PKCS #11 module communicates any error status synchronously through the use of its documented return and reason codes. It is the responsibility of the calling application to handle exceptional conditions in a FIPS 140-2 appropriate manner.

ICSF PKCS #11 is optimized for library use and does not contain any terminating assertions or exceptions. Any internal error detected by ICSF PKCS #11 and not induced by user data will be reflected back to the application with appropriate return and reason codes. The calling application must examine the return and reason codes and act in a FIPS 140-2 appropriate manner to such failures and reflect this error in a fashion consistent with this application.

User-induced or internal errors do not reveal any sensitive material to callers. Return and reason codes and error conditions are fully documented in the product's programming documentation.

Roles, Services and Authentication

Roles

The module supports the definition of multiple virtual PKCS #11 tokens. For each token, the module supports six roles, three for the Security Officer (SO) and three for the User (Table 4). Each of the roles is authenticated through the operating system prior to using any system services. Role identification is implied by the service being requested and the parameters specified. The module does not support explicit user authentication, but does query the operating system to check the user's permissions prior to granting access to the service.

Table 4 Token Role Descriptions and Authentication Mechanisms

Role	Purpose / Permitted Actions	Type of Authentication	Authentication Data	Strength of Mechanism
Weak SO	Read, create, delete, modify, and use public objects	None(Automatic)	None	N/A
SO R/W	Weak SO actions plus initialize token	None(Automatic)	None	N/A
Strong SO	SO R/W actions plus create and delete (but not use) private objects	None(Automatic)	None	N/A
User R/O	Read and use public and private objects	None(Automatic)	None	N/A
Weak User	User R/O actions plus create, delete, modify, and use public and private objects (cannot add, delete, or modify certificate authority objects)	None(Automatic)	None	N/A
User R/W	Weak User actions plus add, delete, and modify certificate authority objects	None(Automatic)	None	N/A

In addition to the above roles, there is one over-all Administrator role, which is for module installation and configuration and for setting the token permissions. This role does not involve the use of cryptographic services.

The Administrator role is not explicitly authenticated but assumed implicitly on implementation of the module's installation and usage sections defined in the security rules section.

Services

The module provides commands and queries (Tables 5 and 6). Queries return status of commands or command groups; commands exercise cryptographic functions. Administrators perform queries; Security Officers and Users may perform both queries and commands as permitted by table 4. While most test queries are not executed automatically as part of regular operations, certain test queries are executed automatically; these special cases are parenthesized as (yes) in Table 6.

Services are accessed through documented API interfaces from the calling application. For a list of API services available in the ICSF PKCS #11 module, see Table 7.

Table 5 Commands

Service	**Notes**	**Modes**	**Approved If yes, Cert #**
Software			
Symmetric Algorithms			
AES	128, 192 or 256 bit keys (FIPS 197)	CBC, ECB	Yes Cert # 1866
AES	128, 192 or 256 bit keys	GCM	Yes Cert # 1866
Triple-DES	168 bit keys	CBC	Yes Cert # 1212
Public Key Algorithms			
DSA Key/Parameter Generation	1024 bit prime w/160 subprime, 2048 bit prime w/224 bit subprime, or 2048 bit prime w/256 bit subprime	N/A	Yes Cert # 584
DSA Sign	1024, 2048 bit prime[1]	N/A	Yes Cert # 584
DSA Verify	1024, 2048 bit prime	N/A	Yes Cert # 584
RSA Key Generation	ANSI X9.31 (1024 to 4096 bits)	N/A	Yes Cert # 949
RSA Sign	PKCS #1 (1024 to 4096 bits)[1]	N/A	Yes Cert # 949
RSA Verify	PKCS #1 (1024 to 4096 bits)	N/A	Yes Cert # 949
RSA Verify for module integrity	PKCS #1 using SHA-256 (IRRPVERS)	N/A	Yes Cert # 946
RSA Encrypt/Decrypt	PKCS #1 (1024 to 4096 bits)	N/A	No
Diffie-Hellman (DH) Parameter Generation	1024, 2048 bit prime	N/A	No
DH Key Generation	1024 - 2048 bits modulus in increments of 64 bits	N/A	Yes Cert # 9
DH Key Agreement	1024 - 2048 bits modulus in increments of 64 bits[3]	N/A	Yes Cert # 9
EC DH Key Generation	NIST P-192 – P-521	N/A	Yes Cert # 9
EC DH Key Agreement	NIST P-192 – P-521[4]	N/A	Yes Cert # 9
ECDSA Key Generation	NIST P-192 – P-521	N/A	Yes Cert # 261
ECDSA Sign	NIST P-192 – P-521[2]	N/A	Yes Cert # 261
ECDSA Verify	NIST P-192 – P-521	N/A	Yes Cert # 261
Hash Functions			
SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	FIPS 180-3	N/A	Yes Cert # 1641
Message Authentication Codes (MACs)			

Service	**Notes**	**Modes**	**Approved If yes, Cert #**
HMAC for SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	Key sizes greater or equal to ½ the output hash size (FIPS 198, 198a)	N/A	Yes Cert # 1112
HMAC-MD5	Part of TLS specific service	N/A	No
DRBG	NIST SP 800-90	N/A	Yes Cert # 151
CP Assist for Cryptographic functions			
Symmetric Algorithms			
AES	128, 192 or 256 bit keys (FIPS 197)	CBC, ECB	Yes Cert # 1713
Triple-DES	168 bit keys	CBC, ECB	Yes Cert # 1103
Hash Functions			
SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	FIPS 180-3	N/A	Yes Cert # 1497
4765-001 (CEX3A) Hybrid			
Public Key Algorithms			
RSA Encrypt/Decrypt	PKCS #1 (1024 to 4096 bits) PKCS #1 block formatting performed in software. CEX3A used for RSA acceleration only	N/A	No
RSA Sign	PKCS #1 (1024 to 4096 bits)[1] PKCS #1 block formatting performed in software. CEX3A used for RSA acceleration only	N/A	Yes Cert # 971
RSA Verify	PKCS #1 (1024 to 4096 bits) PKCS #1 block formatting performed in software. CEX3A used for RSA acceleration only	N/A	Yes Cert # 971
Other functions present but not allowed in FIPS mode			
DES Encrypt/Decrypt	Software and CPACF	N/A	No
2-key Triple-DES	CPACF	N/A	No
RC4 Encrypt/Decrypt	Software	N/A	No
BLOWFISH Encrypt/Decrypt	Software	N/A	No
MD2	Software	N/A	No
MD5	Software	N/A	No
RIPE-MD	Software	N/A	No
RSA 512-1024 bit keys	Software	N/A	No
DSA 512-1024 bit keys	Software	N/A	No

Service	Notes	Modes	Approved If yes, Cert #
DH 512-1024 bit keys	Software	N/A	No
EC Brainpool curves	Software	N/A	No
HMAC key sizes less than ½ output hash size	Software	N/A	No

Notes

1. Use of DSA or RSA private keys less than 2048 bits in length for digital signature generation is deprecated and should not be used after 2013.

2. Use of ECDSA private keys less than 224 bits in length for digital signature generation is deprecated and should not be used after 2013.

3. Use of DH keys less than 2048 bits in length for key agreement is deprecated and should not be used after 2013.

4. Use of ECDH keys less than 224 bits in length for key agreement is deprecated and should not be used after 2013.

Table 6 Queries

Service	**Notes**	**Roles**		
Module Status		**Admin**	**SO**	**User**
Query mode	Check if module is in FIPS 140-2 mode (CCVTFIPS and CCVTFIPS_COMPAT)	Yes	Yes	Yes
Integrity Checks				
Power-up Tests	Automatic before first use	(yes)	No	No
Self-Tests	CSFINPV2 self-test requires restarting ICSF IRRPVERS self-test requires system IPL	Yes	n/a	n/a
Operational Correctness Checks				
RNG Tests	Continuously performed (automatic)	n/a	Yes	Yes
Pair-wise consistency	Continuously performed (automatic)	n/a	Yes	Yes

Operational Environment

Installation and Invocation

RACF level HRF7780 is installed as part of the z/OS Version 1 Release 13 ServerPac using the “Installing Your Order” documentation provided with the ServerPac (prepackaged tailored z/OS installation including RACF). The evaluated configuration requires the installation of additional service provided through APAR OA30951.

ICSF level HCR7780 is shipped as a web deliverable, which may be obtained from the z/OS Downloads web site, http://www.ibm.com/systems/z/os/zos/downloads/. The evaluated configuration requires the installation of additional service provided through APAR OA36882.

After installation, ICSF must be started to make the ICSF PKCS #11 module available. Prior to starting, the Administrator must specify the desired FIPSMODE option in the ICSF options data set:

FIPSMODE(YES,FAIL(YES|NO))

- FIPS standard mode – Provides full FIPS support. All applications must adhere to FIPS restrictions.

FIPSMODE(COMPAT,FAIL(YES|NO))

- FIPS compatibility mode – Provides FIPS support for select applications. Applications that are not “FIPS Exempt” must adhere to FIPS restrictions.

These options are fully described in the programming documentation.

The cryptographic modules are invoked via the APIs, as documented in the programming documentation.

The hardware accelerator (CEX3A) is installed and configured by an IBM customer engineer (CE) in accordance with the card's manual and installation procedures.

Module Operation

The ICSF PKCS #11 security module is written mostly in C and PL/X, with certain functionality contained within assembler, such as functions that utilize the CPACF. Extensive internal consistency checks verify both user input and module configuration, terminating early if errors are encountered. Internal errors are externalized and do not terminate execution, since the code has been developed mainly for library use.

Since ICSF PKCS #11 can access certain platform-specific functionality, which is not represented in higher-level languages, ICSF PKCS #11 uses a mixture of high-level and platform-specific native code.

The ICSF PKCS#11 security module is intended to operate within the z/OS Version 1, Release 13 in a single-user mode of operation.

Using z/OS ICSF PKCS #11 in a FIPS 140-2 approved manner assumes that the following defined criteria are followed:

- The Operating System enforces authentication method(s) to prevent unauthorized access to Module services.
- All host system components that can contain sensitive cryptographic data (main memory, system bus, disk storage) must be located within a secure environment.
- The unauthorized reading, writing or modification of the ICSF started task is not allowed.
- The ICSF PKCS #11 setup procedures documented in the programming documentation must be followed and setup done correctly. This includes the setting of the cryptographic module's name as outlined in the *z/OS Cryptographic Services Integrated Cryptographic Service Facility System Programmer's Guide.*
- The ICSF PKCS #11 module must be initialized to execute in the FIPS 140-2 mode of operation. This is accomplished through the FIPSMODE ICSF startup option.
- The CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 must be installed and enabled.
- The Crypto Express3 cards, if present, are installed and operated in accordance with its Security Policy.

This module implements both approved and non-approved services. The calling application controls the invocation of the services and the cryptographic material being supplied or used by the services. In FIPS 140-2 mode, the module only allows approved algorithms to be used. The module also offers non-approved but allowed RSA key establishment and exchange services when operating in FIPS mode. The FIPS 140-2 configuration automatically inhibits parameter combinations that are technically possible, but not permitted in FIPS 140-2 mode.

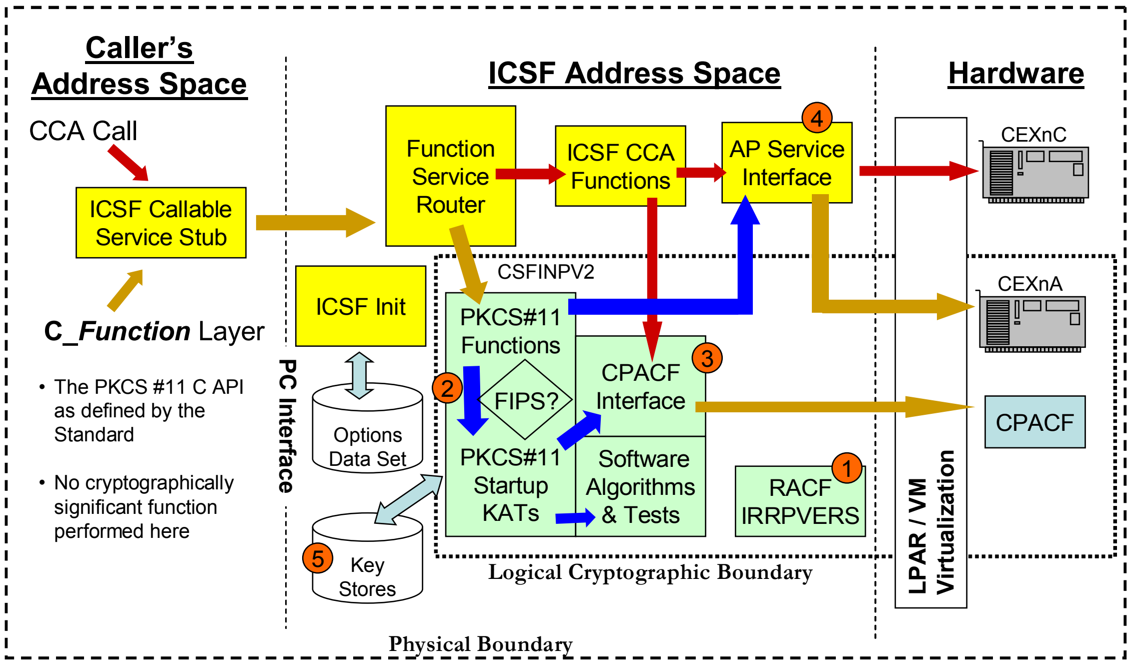
The ICSF PKCS #11 module (CSFINPV2), RACF Program Verification (IRRPVERS), ICSF AP Service Interface (APSI), and CEX3A represents the logical boundary of the module. The physical cryptographic boundary for the module is defined as the enclosure of the host on which the cryptographic module is to be executed.

Although the ICSF APSI is part of the logical cryptographic boundary, it is not deemed as cryptographically relevant to the cryptographic boundary. The ICSF APSI is being treated as a "pipe" between the ICSF PKCS #11 module and the CryptoExpress3 cards. No cryptographic operations are being performed on the data being provided by the ICSF PKCS #11 module until it arrives at the CryptoExpress3 cards.

Conversely, the CPACF Interface does perform cryptographically relevant function and, thus, is physically part of the CSFINPV2 signed program object.

The RACF Signature Verification module (IRRPVERS) is shipped as part of the Security Server RACF FMID. IRRPVERS is the only RACF FIPS cryptographic relevant module, the rest of the RACF FMID is deemed as not being cryptographically relevant. Therefore is not considered part of the cryptographic boundary.
PKCS #11 keys are stored in a dedicated VSAM data set in the clear. Consequently, this keystore and its I/O functions are not considered part of the cryptographic boundary.

As shown in figure 1, ICSF PKCS #11 cryptographic module software components are completely contained within the ICSF started task address space. The CEX3A and CPACF hardware (Hybrid) components reside below the LPAR / VM virtualization layer. ICSF Callable Service stubs, instantiated in the callers address space, exist for each callable service provided by ICSF, both for CCA and PKCS #11. These stubs are merely glue routines that provide no cryptographically relevant function. ICSF also provides a set of DLLs in support of the PKCS #11 standard API. Like the stubs, these DLLs are instantiated in the callers address space and provide no cryptographically relevant function.



🟩	Digitally signed
🟨	Unsigned
➡	FIPS
➡	Non-FIPS
➡	Common
⇔	I/O or data flow

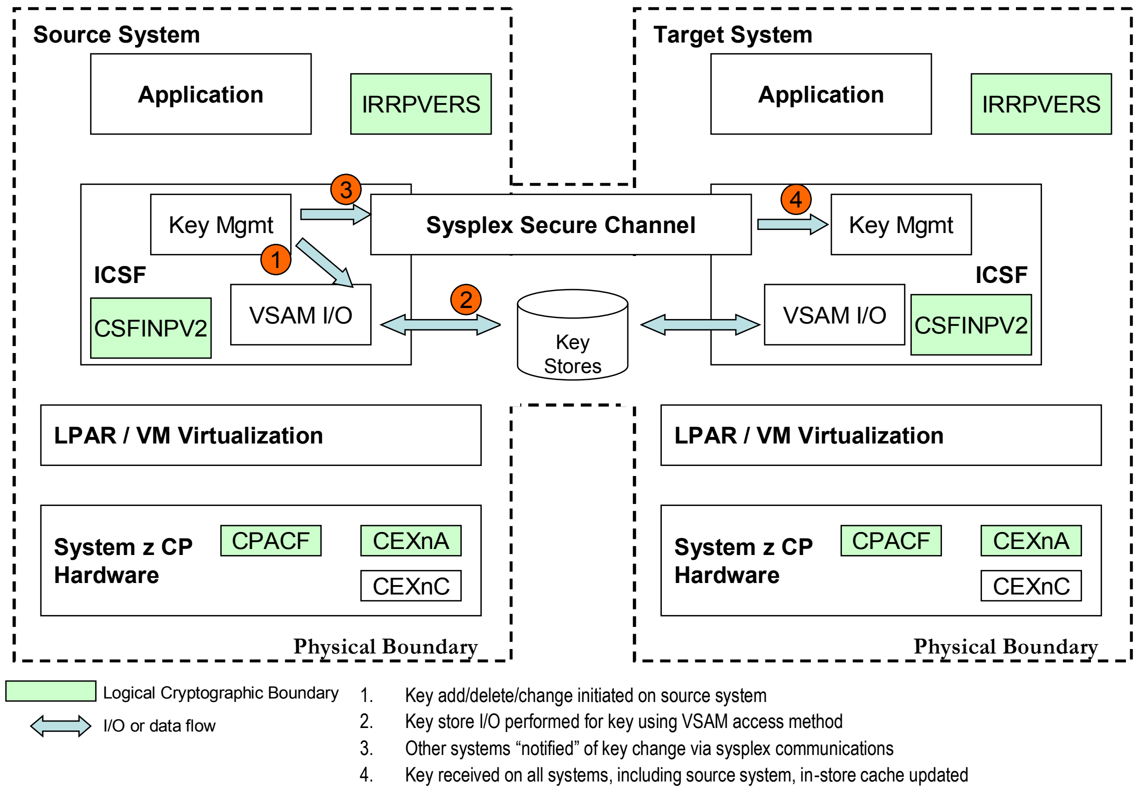
1. RACF Program Verification handles ensuring the integrity of the modules within the cryptographic boundary.
2. PKCS #11 software and CPACF KATs driven at ICSF startup if FIPSMODE option requested
3. CPACF Interface shared by PKCS #11 and CCA services
4. AP Service Interface shared by PKCS #11 and CCA services - "pass-through" only
5. PKCS #11 keys are stored in the clear and thus not considered part of the cryptographic boundary

Figure 1: ICSF PKCS #11 Cryptographic Module

As shown in figure 2, ICSF PKCS #11 cryptographic module may be deployed in a high availability environment where ICSF and other applications may be in effect instantiated on multiple z/OS system instances configured in a "clustered" environment known as a parallel sysplex. A parallel sysplex enables these systems behave like a single, logical computing facility. The underlying structure of the Parallel Sysplex remains virtually transparent to users, networks, applications, and even operations. If the cryptographic module is configured to run in Sysplex mode, ICSF utilizes a secure communication channel provided by the z/OS operating environment to communicate with other instances of ICSF executing on other systems in the sysplex environment.

In a sysplex environment, ICSF synchronizes keys between the individual systems in the sysplex by utilizing z/OS Sysplex Communications. Whenever a key or key attribute is added/deleted/updated on a source system, the target systems are notified of the change via sysplex communications.

Note, that the sysplex secure channel is not considered part of the logical cryptographic boundary as it is just an extension of ICSF's keystore function. However, being that all systems in a sysplex behave like a single, logical computing facility, the physical and logical cryptographic boundaries are extended to include all systems in the sysplex when ICSF is operating in sysplex mode.



1. Key add/delete/change initiated on source system
2. Key store I/O performed for key using VSAM access method
3. Other systems "notified" of key change via sysplex communications
4. Key received on all systems, including source system, in-store cache updated

Figure 2: ICSF PKCS #11 Cryptographic Module in a z/OS Sysplex Environment

Key Management

Key Storage The ICSF PKCS #11 module provides key generation, import and export services to applications such that key material can be used in conjunction with cryptographic services. It is the responsibility of applications using these services to ensure that these services are used in a FIPS 140-2 compliant manner. Keys managed or generated by applications or libraries may be passed from applications to the module in the clear, provided that the sending application or module exists within the physical boundary of the host computer. The module retains such key material within the ICSF address space memory as clear data. In addition, persistent keys are stored in key store data sets (disk storage) as clear data. The disk storage is outside the cryptographic boundary.

Key Generation Key Generation uses an approved RNG algorithm (specified in **NIST SP 800-90**) which is known as Hash_DRBG. It is based on SHA-512 and, thus, has a security strength of 256 bits. All symmetric and asymmetric key generation algorithms use the DRNG engine seeded with 32 bytes of true random data. This true random number generator (TRNG) extracts entropy from either time measurement jitter (minute variations of clock edges) or, if a CryptoExpress3 coprocessor (CEX3C) is installed, data samples from the CEX3C RNG(L) function. The DRNG is reseeded whenever 1M (1,048,576) cumulative bytes have been requested.

Symmetric key generation uses the DRNG directly to produce the key material. DSA and ECDSA key generation is done according to **FIPS Pub 186-3**. When FIPS restricted, RSA key generation implements only the ANSI X9.31 key generation method [3]. A non-compliant RSA key generation method is also present, which allows for the generation of RSA keys shorter than 1024 bits (ANSI X9.31 does not permit generation of shorter keys), but the module does not permit the generation of such short keys when FIPS restricted. DH key generation is similar to DSA key generation. ECDH key generation is similar ECDSA key generation.

Key Entry and Key Exit The module does not support manual key entry or intermediate key generation key output.

Applications may export keys, if desired. Such keys may be exported in-the-clear if the keys are not marked as sensitive. Keys marked sensitive and extractable may be exported in wrapped (encrypted) format. Private keys may be wrapped using 3-key Triple-DES w/CBC PAD or 128, 192, or 256-bit AES w/CBC PAD. Symmetric keys may be wrapped with RSA PKCS #1 v1.5, using any RSA key size from 1024 - 4096 bits.

For **FIPS Pub 140-2**:

1. It is the application's responsibility to choose a wrapping key that is of equal or greater strength than the key being wrapped.
2. It is recommended that applications always mark private and secret keys as sensitive.

Key Establishment When in FIPS 140-2 mode, the module provides support for asymmetric key establishment methods as allowed by Annex D in the **FIPS Pub 140-2**. The supported asymmetric key establishment methods are RSA Key Wrapping, Diffie-Hellman (DH) key agreement, and Elliptic Curve Diffie-Hellman (EC DH) key agreement.

When using Diffie-Hellman in FIPS 140-2 mode, the allowed modulus lengths must be between 1024 and 2048 bits which provides between 80 and 112 bits of encryption strength. Use of a modules length less than 2048 bits is not recommended.

When using Elliptic Curve Diffie-Hellman in FIPS 140-2 mode, the allowed curves are P-192, P-224, P-256, P-384, and P-521 which provide between 96 and 260 bits of encryption strength. Use of the P-192 curve is not recommended.

When using RSA Key Wrapping in FIPS 140-2 mode, the allowed modulus lengths must be between 1024 and 4096 bits which provides between 80 and 150 bits of encryption strength. Use of a modules length less than 2048 bits is not recommended.

Key Protection To enforce compliance with **FIPS Pub 140-2** key management requirements on the ICSF PKCS #11 module, code issuing calls must manage keys in a **FIPS Pub 140-2** compliant method. Keys managed or generated by applications may be passed from the application to the module in the clear in the **FIPS Pub 140-2** validated configuration.

The management and allocation of memory is the responsibility of the operating system. With z/OS, a unique address space is allocated for each started task, such as ICSF. z/OS and the underlying hardware control access to such address spaces. The PKCS #11 module relies on such address space separation to maintain confidentiality of secrets.

Module services and key record storage on disk are protected by z/OS (RACF) access control. The PKCS #11 module relies on this access control to protect against the unauthorized modification, substitution, and use of keys, including public keys.

All keys are associated with the User and SO roles. It is the responsibility of application program developers to protect keys exported from the ICSF PKCS #11 module.

Key Destruction ICSF PKCS #11 objects, when released on behalf of a caller, are not erased by ICSF before they are released. z/OS ensures that the real storage frames that once backed the storage released are zeroed before the frames are reallocated to another caller. The module's administrator may initiate the zeroization of all keys and other secrets managed by ICSF PKCS #11 with the following procedure:

1. Stop the ICSF started task.
2. For persistent key objects stored in DASD key stores, delete the key store data sets. Installations must activate the erase-on-scratch feature to ensure that the DASD storage that backs the key stores is physically erased.

It is the calling application's responsibility to destroy any key objects and similar sensitive information it manages, when no longer needed using **FIPS Pub 140-2** compliant procedures.

Physical Security

The ICSF PKCS #11 installation inherits the physical characteristics of the host running it. The ICSF PKCS #11 module has no physical security characteristics of its own. Figure 3 illustrates an IBM System 196 (z196) mainframe computer.

The CEX3A is a hardware device (see Figure 4) optionally installed to provide hybrid functionality to the System SSL module. In order to meet **FIPS Pub 140-2** requirements they must meet the physical security requirements of Security Level 1. Security Level 1 is satisfied by the device (card) being included within the physical boundary of the module and the device being made of commercial-grade components.

The CP Assist for Cryptographic Function (CPACF) (see Figure 5 and 6) offers the full complement of the Triple-DES algorithm, Advanced Encryption Standard (AES) algorithm and Secure Hash Algorithm (SHA).

CPACF Physical Design: Each two microprocessors (cores) on the quad-core chip share a Co-Processor Unit (CoP), which implements the crypto instructions and also provides the hardware compression function. The compression unit is integrated with the CP Assist for Cryptographic Function (CPACF), benefiting from combining (sharing) the use of buffers and interfaces. The CoP is located on the processor die and is connected to two cores and to L2 cache with dedicated buses.

Figure 3 IBM System z196 Mainframe Computer.

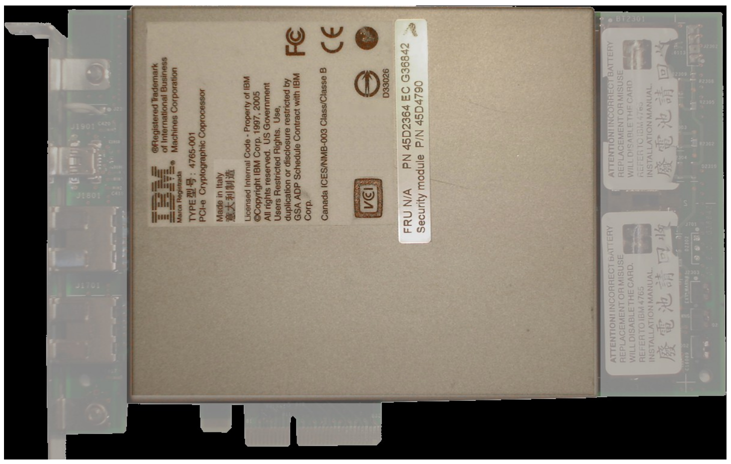


Figure 4 Crypto Express3 Card.

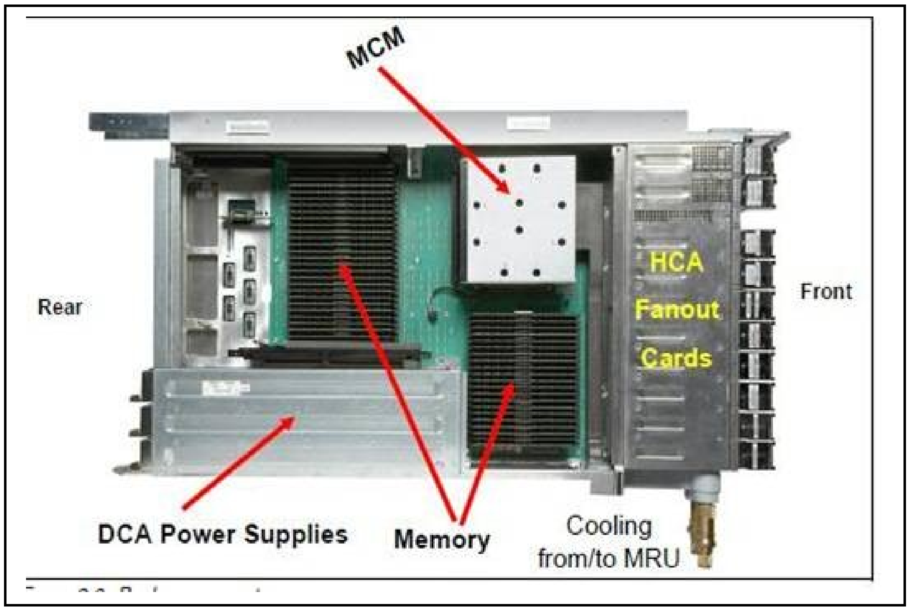
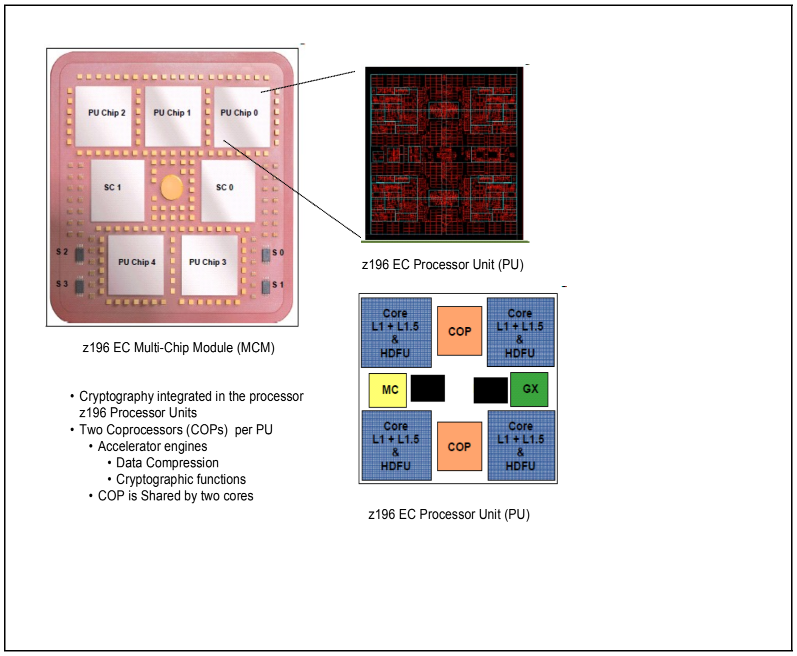


Figure 5 CPBOOK



- Cryptography integrated in the processor z196 Processor Units
- Two Coprocessors (COPs) per PU
  - Accelerator engines
    - Data Compression
    - Cryptographic functions
  - COP is Shared by two cores

Figure 6 Processor Unit MCM with CPACF COP chips

EMI/EMC

EMI/EMC properties of ICSF PKCS #11 are not meaningful for the ICSF PKCS #11 module itself. Systems utilizing the module's services have their overall EMI/EMC ratings determined by the host system. The validation environments have FCC Class A ratings.

EMI/EMC requirements for the CEX3A card are met by the card's FCC Class B rating.

Self-Tests

ICSF PKCS #11 Module

The ICSF PKCS #11 module implements a number of self-tests to check proper functioning of the module including power-up self-tests and conditional self-tests. Conditional tests are performed when symmetric or asymmetric keys are generated. These tests include a continuous random number generator test and pair-wise consistency tests of the generated DSA or RSA keys.

Startup Self-Tests "Power-up" self-tests consist of software integrity test(s) and known-answer tests of algorithm implementations. The module integrity test is automatically performed during loading. The known-answer tests are performed after the loading is complete, during module initialization. If either of these tests fail, the module is rendered unusable.

The integrity of the module is verified by checking an RSA/SHA-256-based digital signature of each module binary prior to being utilized in FIPS 140-2 compliant mode. Initialization will only succeed if all utilized module signatures are verified successfully. Module signatures are generated during the final phase of the build process. The integrity verification involves IRRPVERS verifying its own digital signature. Once verified, IRRPVERS verifies the digital signature of CSFINPV2.

Algorithm known answer tests are performed by module initialization when the **FIPSMODE(YES,…)** or **FIPSMODE(COMPAT,…)** ICSF option has been specified to indicate FIPS 140-2 compliant mode. The tests are performed prior to any cryptographic algorithms being executed in FIPS 140-2 mode.

The module tests the following cryptographic algorithms:

CPACF – AES, Triple-DES, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512.

Software – AES, Triple-DES, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, RSA (sign/verify, encrypt/decrypt), DSA (sign/verify), ECDSA (sign/verify), ECDH, HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, GCM, and the DRNG.

Self-tests are performed in logical order, verifying module integrity incrementally:
1. Integrity test on module, using RSA/SHA-256
2. Known-answer tests on algorithms, from integrity-verified binary.

Once the startup self tests are complete (and successful), the module will issue console message CSFM015I. It will also set either CCVTFIPS or CCVTFIPS_COMPAT depending on the startup option requested.

In addition to the above, the CEX3A also performs its own startup known answer tests for RSA sign/verify.

Startup Recovery If any of the startup self-tests fail, ICSF PKCS #11 will terminate FIPS 140-2 processing.

Conditional Self-Testing Conditional self-testing includes continuous DRNG testing. Continuous DRNG testing involves comparing every newly-generated RNG block with the previously-generated one. The first output block generated by DRNG is used only for the purpose of initiating the continuous DRNG test. The test fails if the DRNG outputs the same value twice subsequently.

If the DRNG outputs identical, subsequent pseudo-random blocks, it enters an error state and returns the corresponding status. The calling application must recognize this error and handle it in a FIPS 140-2 appropriate manner, for example, by reinitializing the module instance.

Similar to the DRNG, high-entropy seed extracted by the TRNG is checked for repeated blocks, before seeding the DRNG. If blocks of entropy repeat, the TRNG reports a failure, which caller applications must also handle as an error.

Pair-wise Consistency Checks This test is run whenever the module generates a DSA, ECDSA, or RSA public/private key-pair.

If the pair-wise consistency check fails, the module enters an error state and returns an error status code. The calling application must recognize this error and handle it in a FIPS 140-2 appropriate manner, for example, by restarting ICSF.

Invoking FIPS 140-2 self-tests on demand. If a user can access ICSF PKCS #11 services, the module has passed its integrity and power-up self tests. The module does not provide a method to drive the self-test on demand, short of restarting ICSF.

A system ipl is required for the RACF IRRPVERS module to repeat the known answer tests on demand.

Crypto Express3

The IBM Crypto Express3 features whether configured as an accelerator or coprocessor executes the following self-tests upon every startup:

A configuration integrity test verifies firmware flash memory modules and code integrity. The initial and continuous checks are basically identical, verifying memory checksums when required. The initial checks verify integrity once before data is used for the first time. Non-modifiable firmware is checked for integrity through embedded checksums. In case of checksum mismatch, the code halts itself or is not even permitted to execute. This code is executed only at startup.

Functional integrity of hardware components is tested through a selected set of known answer tests, covering all programmable components. The programmable devices verify their own code integrity, external tests verify proper connectivity. CPU integrity is verified as part of power on self test before execution continues to load the embedded operating system. These checks verify fundamental functionality, such as proper execution control, load/store operations, register functions, integrity of basic logical and arithmetic operations, and so forth. Once the CPU tests pass, CPU failures are monitored using other error-checking mechanisms (such as parity checks of the PCI bus etc.)

FPGA integrity (communications firmware) is checked by the FPGA itself, through a checksum embedded in the image, upon loading. If the test fails, the FPGA does not activate, and the card remains inaccessible. After initialization, FPGA interfaces and internals are covered through parity checks internally, and external end-to-end checks at higher logical levels. Crypto ASIC integrity is verified by comprehensive known-answer tests at startup, covering all possible control modes. These tests implicitly cover FPGA transport as well, since tests are performed using both available internal interfaces. During regular operations, the crypto ASIC covers all traffic through combinations of redundant implementations, CRCs, and parity checks, in a way specific to each crypto engine. Any failure is indicated as a hardware failure, to the module CPU and the host.

Modular math engine self-tests cover all possible control modes, and different sizes of modular arithmetic. The modular math primitives' testing covers only modular arithmetic, up to full exponentiation, but not algorithm level (i.e., RSA or DSA protocols).

Interactive communications tests verify that the card PCI-X bus is functioning properly. As part of automatic self-tests, critical functions tests cover the module CPU cache control logic (data and instruction), processor registers, and instruction set; PCI-X bus transport integrity (including communication mailboxes), and RAM module integrity.

In addition to startup tests, the Crypto Express3 conditional data tests that are applicable to its use in this security policy are continuous integrity checks on modular math arithmetic (including RSA and DSA operations), implemented in hardware.

To execute the self-tests on demand, the Crypto Express3 cards required a reboot from the hardware management console.

Operational Requirements (Officer/User Guidance)

Module Configuration for FIPS 140-2 Compliance

To verify FIPS 140-2 compliant usage, the following requirements must be observed:

- Administrators and users of ICSF PKCS #11 must verify that the correct Security Manager Profiles have been defined to ensure that startup integrity tests are performed. Each executable (IRRPVERS and CSFINPV2) contains an RSA/SHA-256 signature. The startup integrity tests ensure that the signatures match the expected value. Once the module passes its startup integrity tests, the administrator or user must verify that the module is still in FIPS 140-2 mode. The Administrator would do this by looking for message CSFM015I. The user or SO would do this by programmatically checking the Cryptographic Communication Vector Table (CCVT) flags CCVTFIPS or CCVTFIPS_COMPAT.

- Applications and libraries using ICSF PKCS #11 features must observe **FIPS Pub 140-2** rules for key management and provide their own self-tests.

  For proper operations, the administrator or user must verify that applications comply with this requirement. While details of these application requirements are outside the scope of this policy, they are mentioned here for completeness.

- The Operating System (OS) hosting the module must be set up in accordance with **FIPS Pub 140-2** rules. It must provide sufficient separation between processes to prevent inadvertent access to data of different processes. (This requirement was met for all platforms tested during validation.)

- Applications using ICSF PKCS #11 services must verify that key ownership is not compromised and keys are not shared between different users of the calling application.

  Note that this requirement is not enforced by the ICSF PKCS #11 module itself, but by the application providing the keys to ICSF PKCS #11.

- Applications utilizing ICSF PKCS #11 services must avoid using non-approved algorithms or modes of operation. If not feasible, the applications must indicate that they use non-approved cryptographic services.

- To be in FIPS 140-2 mode, the ICSF PKCS #11 installation must run on a host with commercial grade components and must be physically protected as prudent in an enterprise environment.

- **Physical assumptions**
  - The module is intended for application use in user areas that have physical control and monitoring. It is assumed that the following physical conditions will exist:
  - **LOCATION**
    - The processing resources of the module will be located within controlled access facilities that will prevent unauthorized physical access.
  - **PROTECTION**
    - The module hardware and software critical to security policy enforcement will be protected from unauthorized physical modification.
    - Any sysplex communications shall be configured so that unauthorized physical access is prevented.
- **Personnel assumptions**
  - It is assumed that the following personnel conditions will exist:
  - **MANAGE**
    - There will be one or more competent individuals assigned to manage the module and the security of the information it contains.
  - **NO EVIL ADMINISTRATOR**
    - The system administrative personnel are not careless, willfully negligent, or hostile, and will follow and abide by the instructions provided by the administrator documentation.
  - **CO-OPERATION**
    - Authorized users possess the necessary authorization to access at least some of the information managed by the module and are expected to act in a cooperative manner in a benign environment.

Determining Mode of Operation

The user or SO may determine the module's mode of operation programmatically checking the Cryptographic Communication Vector Table (CCVT) flags CCVTFIPS or CCVTFIPS_COMPAT.

Applications utilizing services must enforce key management compliant with **FIPS Pub 140-2** requirements. This should be indicated in an application-specific way that is directly observable by administrators and end-users.

While such application-specific details are outside the scope of the validation, they are mentioned here for completeness.

In FIPS 140-2 mode, the module automatically restricts algorithms usage to approved or allowed algorithms and inhibits parameter combinations that are technically legal but outside standardized range (such as nonstandard DSA key sizes, short HMAC keys, etc.). Product documentation describes these additional limitations.

Mitigation of Other Attacks

The Mitigation of Other attacks security section of FIPS 140-2 is not applicable to the ICSF PKCS #11 cryptographic module.

Cryptographic Module Configuration Diagrams

The following diagrams illustrate the different validated configurations. These validated configurations can consist of a single z/OS System instance or multiple z/OS System instances.

Figure 7 illustrates the IBM zEnterprise 196 (z196) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 configuration (Base GPC).

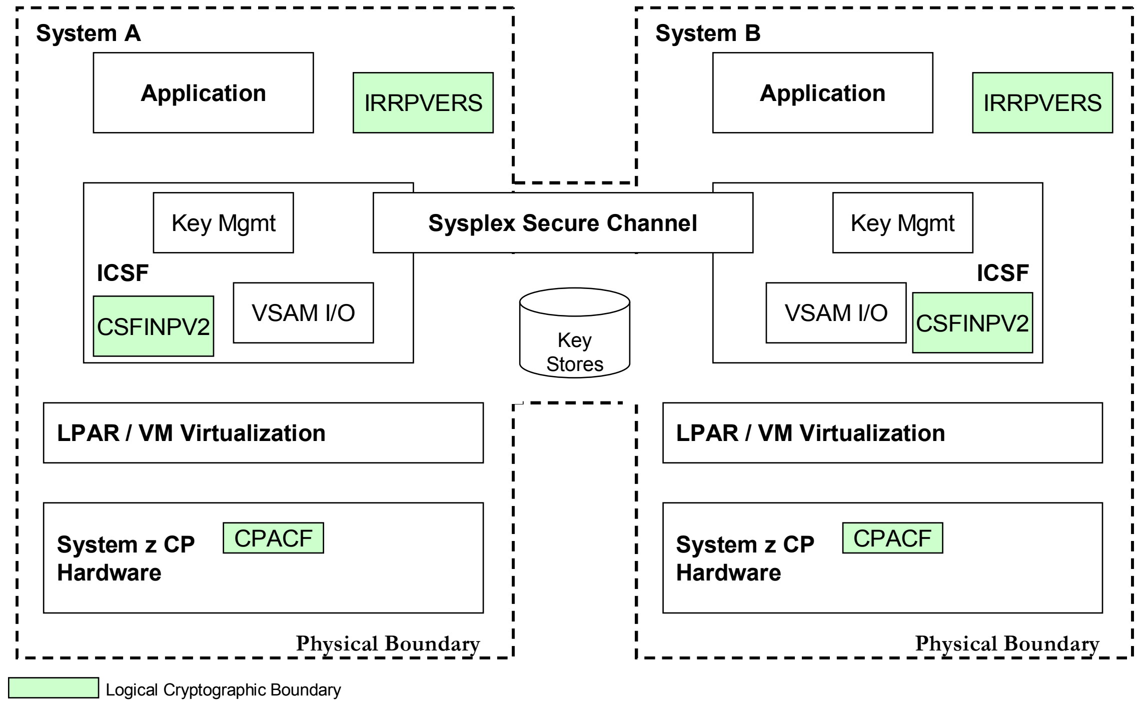


Figure 7 Validated Configuration with CPACF only.

Figure 8 illustrates the IBM IBM zEnterprise 196 (z196) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and Crypto Express3 cards (Accelerator (CEX3A)) configuration.

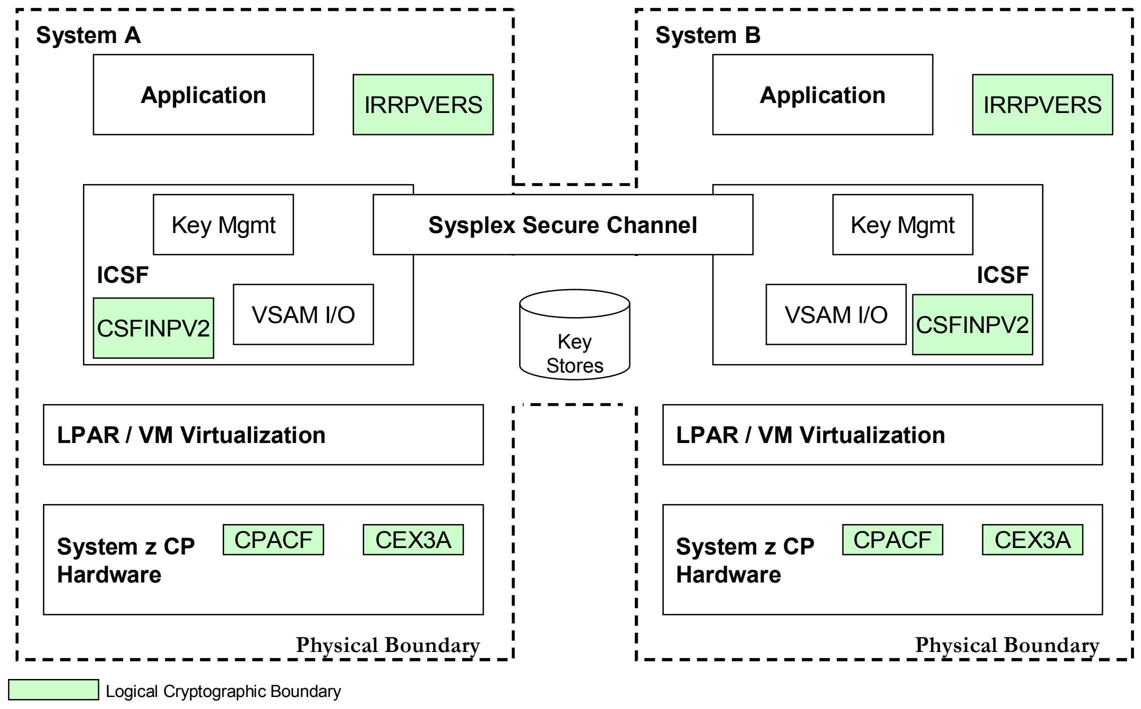


Figure 8 Validated Configuration with CPACF and CEX3A.

Application Programming Interfaces (APIs)

The following Services (APIs) in Table 7 can be executed by the user or SO. The approved/allowed services used by the APIs are:

- Triple-DES, AES
- SHA-1, SHA2 (SHA-224, SHA-256, SHA-384 and SHA-512)
- HMAC-SHA, HMAC-SHA2 (SHA-224, SHA-256, SHA-384 and SHA-512)
- RSA sign/verify, encrypt/decrypt, key generation
- DSA sign/verify, key parameter and key generation
- ECDSA sign/verify, key parameter and key generation
- Diffie-Hellman key agreement, key parameter and key generation
- EC Diffie-Hellman key agreement and key generation
- DRNG

Table 7 ICSF PKCS #11 Module Services (APIs)

Verb	***Service Name***	***Description***	***Called By***
CSFPDMK	PKCS #11 Derive multiple keys	Generate multiple secret key objects and protocol dependent keying material from an existing secret key object Supports mechanisms: CKM_SSL3_KEY_AND_MAC_DERIVE, and CKM_TLS_KEY_AND_MAC_DERIVE Additional vendor mechanisms for IPSec	C_DeriveKey
CSFPDVK	PKCS #11 Derive key	Generate a new secret key object from an existing key object Supports mechanisms: CKM_DH_PKCS_DERIVE, CKM_SSL3_MASTER_KEY_DERIVE, CKM_SSL3_MASTER_KEY_DERIVE_DH, CKM_TLS_MASTER_KEY_DERIVE, CKM_TLS_MASTER_KEY_DERIVE_DH, and CKM_ECDH1_DERIVE Additional vendor mechanisms for IPSec	C_DeriveKey
CSFPHMG	PKCS #11 Generate HMAC	Generate a hashed message authentication code (MAC) Supports mechanisms: CKM_MD5_HMAC, CKM_SHA_1_HMAC, CKM_SHA224_HMAC, CKM_SHA256_HMAC, CKM_SHA384_HMAC, CKM_SHA512_HMAC, CKM_SSL3_MD5_MAC, and CKM_SSL3_SHA1_MAC	C_Sign, C_SignUpdate, C_SignFinal
CSFPGKP	PKCS #11 Generate key pair	Generate an RSA, DSA, Elliptic Curve, or Diffie-Hellman key pair Supports mechanisms: CKM_RSA_PKCS_KEY_PAIR_GEN, CKM_DSA_KEY_PAIR_GEN, CKM_DH_PKCS_KEY_PAIR_GEN, and CKM_EC_KEY_PAIR_GEN	C_GenerateKeyPair

CSFPGSK	PKCS #11 Generate secret key	Generate a secret key or set of domain parameters Supports mechanisms: CKM_DES_KEY_GEN, CKM_DES2_KEY_GEN, CKM_DES3_KEY_GEN, CKM_AES_KEY_GEN, CKM_DSA_PARAMETER_GEN, CKM_DH_PKCS_PARAMETER_GEN, CKM_BLOWFISH_KEY_GEN, CKM_RC4_KEY_GEN, CKM_GENERIC_SECRET_KEY_GEN, CKM_SSL3_PRE_MASTER_KEY_GEN, and CKM_TLS_PRE_MASTER_KEY_GEN	C_GenerateKey
CSFPGAV	PKCS #11 Get attribute value	List the attributes of a PKCS11 object. For token/object management only, provides no cryptographically relevant function	C_GetAttributeValue
CSFPOWH	PKCS #11 One-way hash, sign or verify	Generate a one-way hash on specified text or sign or verify specified text Supports mechanisms: CKM_MD2, CKM_MD5, CKM_SHA_1, CKM_SHA224, CKM_SHA256, CKM_SHA384, CKM_SHA512, CKM_RIPEMD160, CKM_MD2_RSA_PKCS, CKM_MD5_RSA_PKCS, CKM_SHA1_RSA_PKCS, CKM_SHA224_RSA_PKCS, CKM_SHA256_RSA_PKCS, CKM_SHA384_RSA_PKCS, CKM_SHA512_RSA_PKCS, CKM_DSA_SHA1, CKM_DSA_SHA224, CKM_DSA_SHA256, CKM_DSA_SHA384, CKM_DSA_SHA512, CKM_ECDSA_SHA1, CKM_ECDSA_SHA224, CKM_ECDSA_SHA256, CKM_ECDSA_SHA384, and CKM_ECDSA_SHA512	C_Digest, C_DigestUpdate, C_DigestFinal, C_Sign, C_SignUpdate, C_SignFinal, C_Verify, C_VerifyUpdate, C_VerifyFinal
CSFPPKS	PKCS #11 Private key sign	Decrypt or sign data using an RSA private key using zero-pad or PKCS #1 1.5 formatting. Sign data using a DSA private key. Sign data using an Elliptic Curve private key in combination with DSA Supports mechanisms: CKM_RSA_X_509, CKM_RSA_PKCS, CKM_DSA, and CKM_ECDSA	C_Sign, C_SignFinal, C_Decrypt
CSFPPRF	PKCS #11 Pseudo-random function	Generate pseudo-random output of arbitrary length	C_DeriveKey, C_GenerateRandom
CSFPPKV	PKCS #11 Public key verify	Encrypt or verify data using an RSA public key using zero-pad or PKCS #1 1.5 formatting. Verify a signature using a DSA public key. Verify a signature using an Elliptic Curve public key in combination with DSA. Supports mechanisms: CKM_RSA_X_509, CKM_RSA_PKCS, CKM_DSA, and CKM_ECDSA	C_Verify, C_VerifyFinal, C_Encrypt
CSFPSKD	PKCS #11 Secret key decrypt	Decipher data using a clear symmetric key Supports mechanisms: CKM_DES_ECB, CKM_DES_CBC, CKM_DES_CBC_PAD, CKM_DES3_ECB, CKM_DES3_CBC, CKM_DES3_CBC_PAD, CKM_AES_CBC, CKM_AES_ECB, CKM_AES_CBC_PAD, CKM_AES_GCM, CKM_BLOWFISH_CBC, and CKM_RC4	C_Decrypt, C_DecryptUpdate, C_DecryptFinal

CSFPSKE	PKCS #11 Secret key encrypt	Encipher data using a clear symmetric key Supports mechanisms: CKM_DES_ECB, CKM_DES_CBC, CKM_DES_CBC_PAD, CKM_DES3_ECB, CKM_DES3_CBC, CKM_DES3_CBC_PAD, CKM_AES_CBC, CKM_AES_ECB, CKM_AES_CBC_PAD, CKM_AES_GCM, CKM_BLOWFISH_CBC, and CKM_RC4	C_Encrypt, C_EncryptUpdate, C_EncryptFinal
CSFPSAV	PKCS #11 Set attribute value	Update the attributes of a PKCS11 object. For token/object management only, provides no cryptographically relevant function	C_GetAttributeValue
CSFPTRC	PKCS #11 Token record create	Initialize or re-initialize a z/OS PKCS #11 token, creates or copies a token object in the token data set and creates or copies a session object for the current PKCS #11 session. For token/object management only, provides no cryptographically relevant function	C_CreateObject, C_CopyObject, C_InitToken
CSFPTRD	PKCS #11 Token record delete	Delete a z/OS PKCS #11 token, token object, or session object. For token/object management only, provides no cryptographically relevant function	C_DestroyObject, C_InitToken
CSFPTRL	PKCS #11 Token record list	Obtain a list of z/OS PKCS #11 tokens. The caller must have SAF authority to the token. Also obtains a list of token and session objects for a token. Use a search template to restrict the search for specific attributes. For token/object management only, provides no cryptographically relevant function	C_Initialize, C_GetSlotList
CSFPUWK	PKCS #11 Unwrap key	Unwrap and create a key object using another key Supports mechanisms: CKM_RSA_PKCS, CKM_DES_CBC_PAD, CKM_DES3_CBC_PAD, and CKM_AES_CBC_PAD	C_UnwrapKey
CSFPHMV	PKCS #11 Verify HMAC	Verify a hash message authentication code (MAC) Supports mechanisms: CKM_MD5_HMAC, CKM_SHA_1_HMAC, CKM_SHA224_HMAC, CKM_SHA256_HMAC, CKM_SHA384_HMAC, CKM_SHA512_HMAC, CKM_SSL3_MD5_MAC, and CKM_SSL3_SHA1_MAC	C_Verify, C_VerifyUpdate, C_VerifyFinal
CSFPWPK	PKCS #11 Wrap key	Wrap a key with another key Supports mechanisms: CKM_RSA_PKCS, CKM_DES_CBC_PAD, CKM_DES3_CBC_PAD, and CKM_AES_CBC_PAD	C_WrapKey

Glossary

Address space	A set of contiguous virtual addresses available to a program and its data. The address space is a container for enclaves and processes. [4] [5]
API	Application Programming Interface
CEX3A	Crypto Express3 Accelerator, mainframe name for IBM Hardware Security Modules (HSMs).

CEX3C	Crypto Express3 Coprocessor, mainframe name for IBM Hardware Security Modules (HSMs).
CPACF	CP Assist for Cryptographic Function, clear key on-chip accelerator integrated into mainframe processors. CPACF functionality is restricted to symmetric and hashing operations.
DLL	Dynamic Link Library, shared program library instantiated separately from binaries using it. FIPS 140-2 configurations of ICSF PKCS #11 DLLs are never statically linked.
DRNG	Deterministic Random Number Generator, a deterministic function expanding a "true random" seed to a pseudo-random sequence.
Enclave	In the z/OS Language Environment, a collection of routines, one of which is named as the main routine. The enclave contains at least one thread. Multiple enclaves may be contained within a process. [4] [5]
ICSF	Integrated Cryptographic Service Facility
KAT	Known Answer Test
OS	Operating System
Process	A collection of resources; both program code and data, consisting of at least one enclave. [4] [5]
ServerPac	Prepackaged version of the z/OS Operating System
Side deck	The functions and variables that can be imported by DLL applications.
Thread	An execution construct that consists of synchronous invocations and terminations of routines. The thread is the basic run_time path within the z/OS Language Environment program management model, and is dispatched by the operating system with its own run-time stack, instruction counter and registers. Thread may exist concurrently with other threads within an address space. [4] [5]
TRNG	True Random Number Generator, a service that extracts cryptographically-useful random bits from non-deterministic (physical) sources. The "random seed" bits are post-processed by a DRNG.

References

[1] z/OS V1R13.0 elements and features PDF files - Cryptographic Services http://www-03.ibm.com/systems/z/os/zos/bkserv/r13pdf/#crypto with OA34403 APAR documentation ftp://ftp.software.ibm.com/eserver/zseries/zos/icsf/pdf/oa34403.pdf

[2] National Institute of Standards and Technology, Security Requirements for Cryptographic Modules (FIPS 140-2), 2002

[3] American National Standard Institute, Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (X9.31), 1998

[4] ABCs of z/OS System Programming Volume 1 (SG24-6981-01)

[5] ABCs of z/OS System Programming Volume 2 (SG24-6982-02)

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- IBM
- RACF
- z9
- z10
- z/196
- zEnterprise
- z/OS