IBM® z/OS® Version 2 Release 4 ICSF PKCS #11 Cryptographic Module

FIPS 140-2

Non-Proprietary Security Policy

Version 1.4

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

April 6, 2021

Table of Contents

1 Cryptographic Module Specifications

1.1 Scope of Document

This document is the non-proprietary security policy for the IBM® z/OS® Version 2 Release 4 ICSF PKCS #11 Cryptographic Module, and was prepared as part of the requirements for conformance to Federal Information Processing Standard (FIPS) 140-2, Level 1 Software-Hybrid Module.

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].

The software component in which the z/OS ICSF PKCS #11 module is shipped consists of a set of loadable software objects (binary programs and auxiliary files). The deployed version consists of the following objects:

Table 1: ICSF PKCS #11 Module – Software Component

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

The z/OS ICSF PKCS #11 install package consists of the core program (CSFINPV2) that is utilized while operating in FIPS 140-2 mode, as well as some auxiliary programs and files. The auxiliary programs provide functionality that is not cryptographically relevant (i.e. pass-through support). The remaining files consist of headers, side decks, sample application, and sample configuration scripts.

1.2 Cryptographic Module Specification

The z/OS ICSF PKCS #11 module is classified as a multi-chip standalone software-hybrid module for FIPS140-2 purposes. The actual cryptographic boundary for this FIPS 140-2 module validation includes the ICSF PKCS #11 module running in configurations supplemented by hardware cryptography. The ICSF PKCS #11 module consists of software-based cryptographic algorithms as well as symmetric, hashing, true, and deterministic random number generation algorithms provided by the CP Assist for Cryptographic Function (CPACF), and optionally one IBM Cryptographic Coprocessor configured as an accelerator for RSA modular cryptography. The accelerator can be one IBM 4768 Cryptographic Coprocessor (CEX6S) configured as an accelerator (CEX6A).

The CEX6A accelerator is accessed through the auxiliary program CSFINPVT. The ICSF PKCS #11 module interfaces with CSFINPVT which acts as a "pipe" between ICSF PKCS #11 and the cryptographic cards.

The ICSF PKCS #11 module is bound to the z/OS Version 2 Release 4 Security Server RACF® Signature Verification (hereafter referred to as "IRRPVERS") with FIPS 140-2 certificate #2691. IRRPVERS performs the module integrity checking services.

ICSF PKCS #11 testing was performed using the z/OS Version 2 Release 4 operating system with the following platform configurations:

1. IBM z14® with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 (Base GPC)

2. IBM z14 with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 (Base GPC) and IBM 4768 Cryptographic Coprocessor configured as a Crypto Express6 accelerator (CEX6A).

The module running on the above platforms met all FIPS140-2 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 (note that no claim can be made as to the correct operation of the module or the security strengths of the generated keys when porting the module to one of those environments):

1. IBM z14 with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 (Base GPC) and IBM 4767 Cryptographic Coprocessor configured as a CryptoExpress5 accelerator (CEX5A)
2. IBM z13® with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 (Base GPC)
3. IBM z13 with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and Crypto Express5 accelerator (CEX5A)
4. IBM z13 with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and Crypto Express5 coprocessor (CEX5C).

Figure 1 below shows the physical boundary of the System z machine as well as the logical boundary of the module. A more detailed view is shown in the Operational Environment section.

Figure 1: ICSF PKCS #11 Cryptographic Module Physical and Logical Boundaries

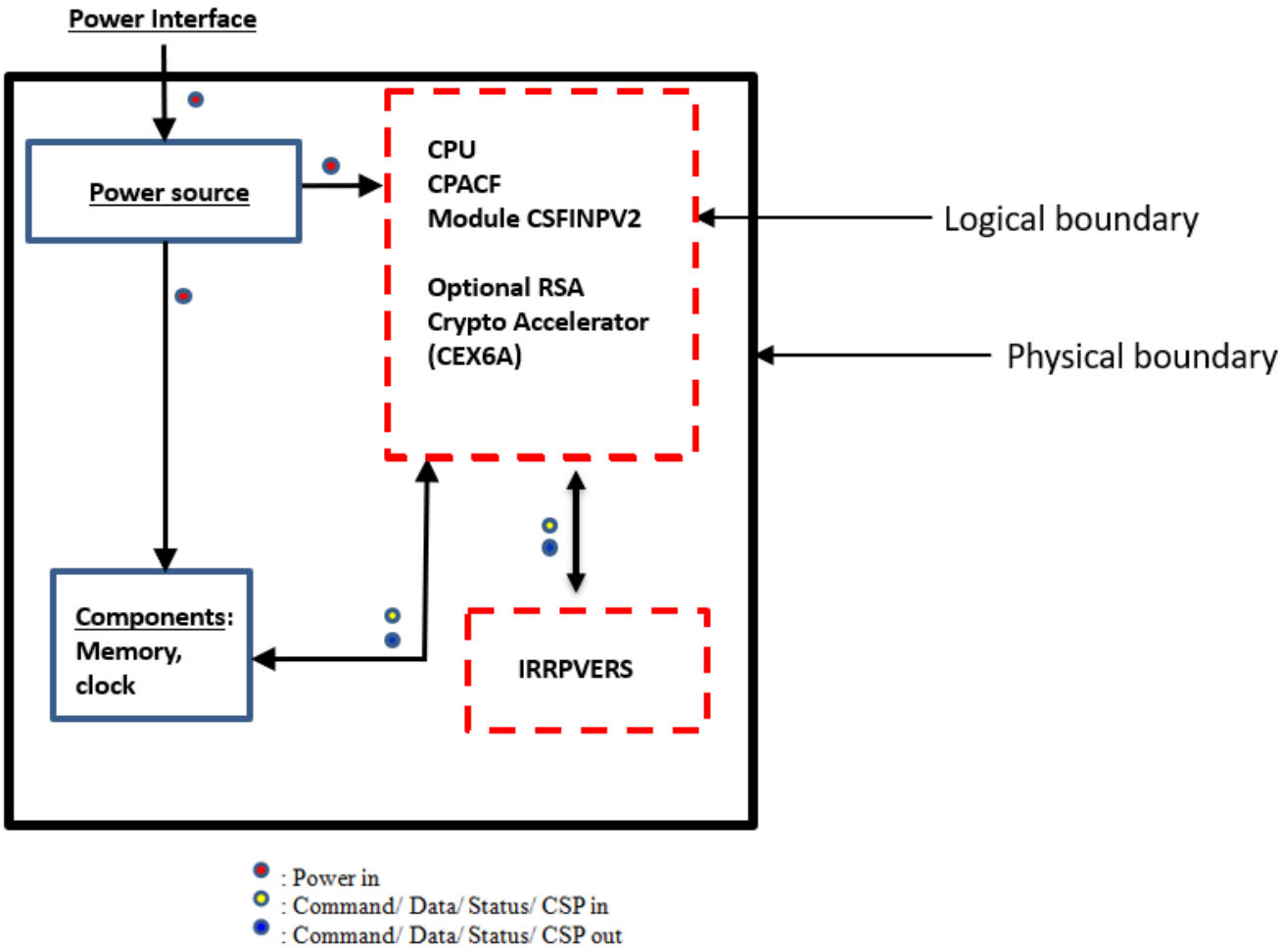


Table 2 lists the module's software and supplementing hardware components and details the versions required and associated documentation.

Table 2: ICSF PKCS #11 Module Components

Type	Names	Version
Software components	ICSF CSFINPV2	ICSF level HCR77D0 with APAR OA58593
Hardware components	CPACF	Firmware – CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 (aka FC3863) with System Driver Level 32L Hardware – COP chips integrated within processor unit
	Crypto Express6 (CEX6S) configured as an accelerator (CEX6A) (4768-001)	Firmware – CCA 6.0.8z Hardware – P/N 01PP167
Documentation		• SC14-7506-08 *z/OS Cryptographic Services ICSF Administrator's Guide* • SC14-7508-08 *z/OS Cryptographic Services ICSF Application Programmer's Guide* • SC14-7509-07 *z/OS Cryptographic Services ICSF Messages* • SC14-7505-08 *z/OS Cryptographic Services ICSF Overview* • SC14-7507-08 *z/OS Cryptographic Services ICSF System Programmer's Guide* • SC14-7511-09 *z/OS Cryptographic Services ICSF TKE Workstation User's Guide* • SC14-7510-06 *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications* • IBM z14 Hardware Management Console online help for the Hardware Management Console

1.3 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

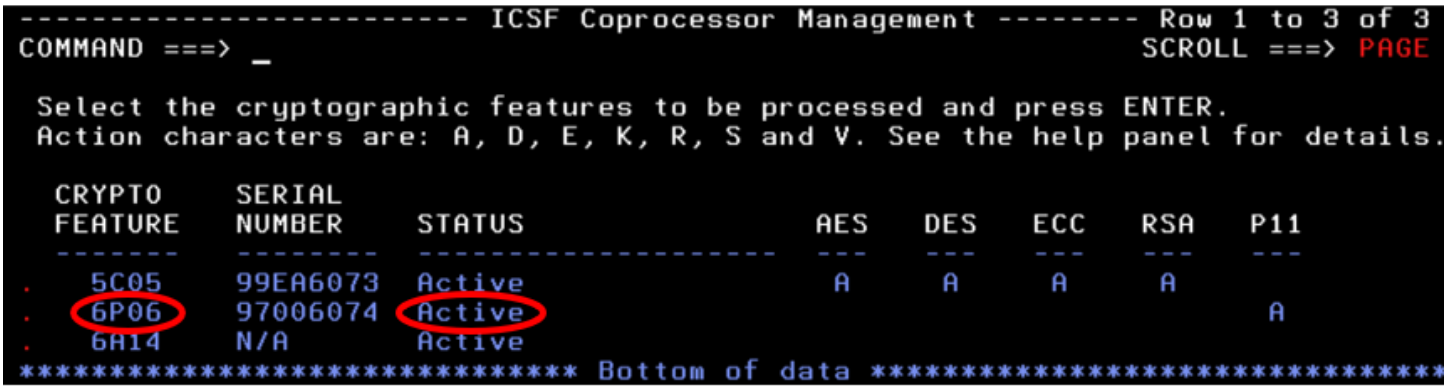
Security Requirements Section	Level
Cryptographic Module Specification	1
Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	1
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of other attacks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

1.4 Determining the Mode of Operation

The ICSF PKCS #11 module offers both FIPS approved and non-approved modes. To ensure FIPS mode operations, user applications must explicitly request FIPS adherence and must only use services (algorithms) defined in tables 6 and 7. The module will temporarily enter non-approved mode when a non-approved algorithm is requested. Thus the mode of operation can be determined by examining the parameters used on the service requests.

Additionally, even if adhering to the requirements for FIPS approved mode, running on a platform configuration that includes an active IBM Crypto Express adapter configured with the Enterprise PKCS #11 (EP11) firmware invalidates the mode. The user or Crypto Officer may check for, and potentially correct, the configuration using the ICSF Coprocessor Management ISPF panel (option 1 from the ICSF main panel). Figure 2 shows a rendering of the panel:

Figure 2: Coprocessor Management Panel



Look for a CRYPTO FEATURE with the prefix “#P”, for example '6P’, that has a STATUS of “Active.” If found, this is a non-approved configuration. The configuration may be corrected by deactivating the coprocessor. This is done by entering a “D” in the left most field on the row and pressing “Enter” key.

Users (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 Crypto Officers and end-users.

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

The user application is responsible for ensuring that only FIPS approved algorithms and key sizes are utilized. User applications must also comply with the key size and algorithm requirements specified in the NIST Special Publication 800-131A Revision 2.

2 Ports and Interfaces

As a multi-chip standalone module, the ICSF PKCS #11 physical interfaces are the boundaries of the host running the ICSF PKCS #11 module code. The underlying logical interfaces of the module are the PKCS #11 callable services documented in the *z/OS Cryptographic Services Integrated Cryptographic Service Facility Application Programmer's Guide* and the ICSF internal API to the Crypto Express Accelerators known as the ICSF Adjunct Processor Service Interface (APSI).

Table 4: Data input, data output, control input and status output

FIPS 140-2 Interface	Logical Interface	Description
Data Input	ICSF PKCS #11 API, Key data sets, ICSF internal API APSI	Input variables are passed on the ICSF PKCS #11 API. Existing keys may be input from key stores (key data sets). Input data is passed via the ICSF internal API APSI to the Crypto Express Card(s).
Data Output	ICSF PKCS #11 API, Key data sets, ICSF internal API APSI	Output results are passed back through the ICSF PKCS #11 API. New keys may be stored back into the key data sets. Output data from the Crypto Express Card(s) is passed to the ICSF PKCS #11 functions via the ICSF internal API APSI.
Control Input	ICSF PKCS #11 API, Options data set	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
Status Output	API	Status output is provided in return and reason codes and through messages
Power	Power Supply, Batteries	Batteries are a backup power source for Crypto Express6 processors

The 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 Guide*.

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

2.1 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.

3 Roles, Services and Authentication

3.1 Roles

The module supports the definition of multiple virtual PKCS #11 tokens. For each token, the module supports a User role – handling requests submitted by either the PKCS #11 Security Officer[1] (SO) or the end-user application (USER). The module implicitly supports a Crypto Officer role as well (see Table 5). Role assumption 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 5: Token Role Descriptions and Authentication Mechanisms

Role	Purpose / Permitted Actions	Type of Authentication	Authentication Data	Strength of Mechanism
User (USER or SO)	Request the cryptographic algorithms listed in tables 6 and 7	Implicit (module checks user's permissions)	None	N/A
Crypto Officer	Module installation and configuration and for setting the token permissions. This role does not involve the use of cryptographic services.	Implicit[2]	N/A	N/A

3.2 Services

The module provides services listed in tables 6 – 8.

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 11.

Additional services are provided by bound module IRRPVERS. This module utilizes the module integrity checking service provided by IRRPVERS.

[1] See the PKCS #11 Cryptographic Token Interface Base Specification Version 2.40 [7] for a description of the PKCS #11 Security Officer. For the purposes of this Security Policy, the PKCS #11 Security Officer falls under the module's User role.

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

Table 6: Approved services in FIPS-Approved mode of operation

Service	***Algorithm***	***Roles***	***Standard***	***CSPs / Caveat***	***Key Lengths, Curves or Moduli***	***Mode / Method***	***Access***	***CAVS Cert #***
Module installation	N/A	Crypto Officer	N/A	N/A	N/A	N/A	N/A	N/A
Query Mode (Show Status)	N/A	User	N/A	N/A	N/A	Check which FIPS 140-2 mode ICSF is running in (CCVTFIPS and CCVTFIPS_ COMPAT)	N/A	N/A
Power-Up Self-Tests	N/A	User	N/A	N/A	N/A	N/A	N/A	N/A
Zeroization	N/A	User	N/A	All keys and CSPs	N/A	N/A	W	N/A
Software								
Symmetric Key Algorithms								
Data Encryption / Decryption	AES	User	SP800-38A-addendum	AES Key	128, 192, or 256 bits	AES-CBC-CS1 (Ciphertext Stealing)	R	(vendor affirmed)
			SP800-38D			GCM		C1635
Key wrapping (when encrypted using the GCM mode of AES)			SP 800-38F			GCM		
Public Key Algorithms								
Parameter Generation	DSA	User	FIPS 186-4	None	L=2048 N=256	N/A	R, W	C1635
Key Generation				DSA Asymmetric private key	L=2048 N=224			
Digital Signature generation				DSA Asymmetric private key	L=2048 N=256		R	
Digital Signature verification				None	L=1024, N=160 L=2048 N=224 L=2048 N=256		N/A	
Key Generation	RSA	User	FIPS 186-4	RSA Asymmetric private key	2048 and 3072 bits	N/A	R, W	C1635
Digital Signature generation (PKCS#1 v1.5 and PSS)			FIPS 186-4		2048 and 3072 bits		R	
			FIPS 186-2		2048, 3072 and 4096 bits			
Digital Signature			FIPS 186-4	None	1024, 2048, 3072 bits		N/A	

Service	*Algorithm*	*Roles*	*Standard*	*CSPs / Caveat*	*Key Lengths, Curves or Moduli*	*Mode / Method*	*Access*	*CAVS Cert #*
verification (PKCS#1 v1.5)			FIPS 186-2		1024, 2048, 3072 and 4096 bits			
Digital Signature verification (PSS)			FIPS 186-4		1024, 2048, 3072 bits			
ECDSA Key Generation	ECDSA	User	FIPS 186-4	ECDSA Asymmetric private key	P-224, P-256, P-384, P-521	N/A	R,W	C1635
Digital Signature generation							R	
Digital Signature verification				None	P-192, P-224, P-256, P-384, P-521		N/A	
Shared secret computation (CVL)	Diffie-Hellman	User	SP 800-56A	Diffie-Hellman keys and shared secret	2048 bits	N/A	R	CVL C1635
	EC Diffie-Hellman	User	SP 800-56A	EC Diffie-Hellman keys and shared secret	P-224, P-256, P-384, P-521	N/A	R	CVL C1635
Hash Functions								
Message Digest	SHS	User	FIPS 180-4	None	N/A	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	N/A	C1635
Message Authentication Codes (MACs)								
Message Authentication	HMAC	User	FIPS 198-1	HMAC key	Key sizes greater or equal to ½ the output hash size[3]	HMAC with SHA-1, SHA-224, SHA-256, SHA-384, or SHA-512	R	C1635
Message Authentication	AES	User	SP 800-38D	AES Symmetric key	128, 192 or 256 bits	GMAC	R	C1635
Random Number Generation								
Random Bit Generation (only used if CPACF Driver 32L not present)	DRBG	User	SP 800-90A	Entropy input, Seed, V, C. Only used if CPACF driver 32L not present.	N/A	SHA-512 Hash DRBG	R,W	C1635
CKG								

[3] Per FIPS 198-1 and SP 800-107, keys less than 112 bits in length are not approved for HMAC generation.

Service	*Algorithm*	*Roles*	*Standard*	*CSPs / Caveat*	*Key Lengths, Curves or Moduli*	*Mode / Method*	*Access*	*CAVS Cert #*
Cryptographic Key Generation	N/A	User	SP 800-133	• AES key • Triple-DES key • HMAC key • RSA private key • DSA private key • DH private key • ECDSA/ECDH private key	• 128, 192, 256 bits • 168 bits • > 112 bits	N/A	R,W	(vendor affirmed)
CP Assist for Cryptographic Functions								
Symmetric Algorithms								
Data Encryption / Decryption	AES	User	FIPS 197 SP 800-38A SP 800-38D	AES Symmetric key	128, 192 or 256 bits	CBC, ECB, CTR	R	C79
Key wrapping when encrypted using the encrypted (but unauthenticated) mode of AES and then authenticated with an GMAC	AES	User	SP 800-38F	AES symmetric key	128, 192 or 256 bits	AES-CBC, AES-ECB or AES-CTR with AES GMAC	R	AES Cert. C79 AES Cert. C79 and C1635
Key wrapping when encrypted using the encrypted (but unauthenticated) mode of AES and then authenticated with an HMAC	AES	User	SP 800-38F	AES symmetric key	128, 192 or 256 bits	AES-CBC, AES-ECB or AES-CTR with HMAC	R	AES Cert. C79 HMAC Cert. C1635
Data Encryption / Decryption	Triple-DES	User	SP 800-67	Triple DES Symmetric key	168 bits	CBC, ECB	R	C79
Message Authentication Codes (MACs)								
Message Authentication	AES	User	SP 800-38D	AES Symmetric key	128, 192 or 256 bits	GMAC	R	C79
Hash Functions								
Message Digest	SHS	User	FIPS 180-4	None	N/A	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	N/A	C79
Random Number Generation								

Service	*Algorithm*	*Roles*	*Standard*	*CSPs / Caveat*	*Key Lengths, Curves or Moduli*	*Mode / Method*	*Access*	*CAVS Cert #*
Random Bit Generation (if CPACF driver 32L not present, used for seeding the software DRBG only)	DRBG	User	SP 800-90A	Entropy input, Seed, V and C	N/A	SHA-512 Hash DRBG	R,W	C1633
4768-001(CEX6A)								
Public Key Algorithms								
Diffie-Hellman shared secret computation	Diffie-Hellman (CVL)	User	SP 800-56A	Diffie-Hellman Asymmetric private key and shared secret	2048 bits	FFC	R	C1637
Digital Signature generation	RSA	User	FIPS 186-4	RSA Asymmetric private key	2048 and 3072 bits	N/A	R	C1637
			FIPS 186-2		4096 bits			
Digital Signature verification			FIPS 186-4	None	1024, 2048 and 3072 bits	N/A	N/A	C1637
			FIPS 186-2		1024, 2048, 3072 and 4096 bits			
Bound IRRPVERS Module								
Digital Signature verification	RSA	User	FIPS 186-4	None	2048 bits	N/A	N/A	C1634

Notes:

1. Use of SHA1 for digital signature generation is deprecated and should not be used.
2. PKCS #1 block formatting performed in software. CEX6A used for RSA acceleration only.
3. The certificate #C79 includes testing of extra algorithms/modes but the only the algorithms mentioned in the above table are used by the module.

Table 7: Allowed services in FIPS-Approved mode of operation

Service	*Algorithm*	**Role**	*Standard*	*CSPs*	*Access*	*Caveat*
Software						
Symmetric Algorithms						
Key unwrapping	AES	User	N/A	AES key	R	128, 192, or 256-bit AES w/CBC PAD
Key unwrapping	Triple-DES	User	N/A	Triple-DES key	R	3-key Triple-DES w/CBC PAD
Public Key Algorithms						

Service	*Algorithm*	Role	*Standard*	*CSPs*	*Access*	*Caveat*
Key wrapping	RSA	User	N/A	RSA private key	R	key wrapping; key establishment methodology provides between 112 and 149 bits of encryption strength; non-compliant less than 112 bits of encryption strength The modulus size at least 2048 bits and up to 4096 bits
Random Number Generation						
Seeding for the DRBGs	NDRNG	User	N/A	seed	R	Implicitly driven
Message Authentication Codes (MACs)						
Message Authentication	MD5 used in HMAC in the context of TLS protocol	User	IETF RFC 2104	HMAC key	R	Allowed in TLS PRF only.
4678-001(CEX6A)						
Public Key Algorithms						
Key Wrapping	RSA	User	N/A	RSA private key	R	The modulus size at least 2048 bits and up to 4096 bits

Table 8: Non-approved services (deprecated algorithms by NIST SP 800-131A or non-compliant) in non-FIPS-Approved mode of operation

Service	*Role*	*Algorithm*	*Notes*
CP Assist for Cryptographic Functions			
Key wrapping	User	AES (using ECB, CBC, CTR modes)	FIPS 197, SP 800-38A
Key Wrapping	User	Triple-DES (using ECB, CBC, CTR modes)	SP 800-67, SP 800-38A
Software			
Public Key Algorithms			
Key generation / Digital signature generation Key Wrapping	User	RSA	Key bit sizes less than 2048 not approved
Parameter generation / Key generation / Digital signature generation	User	DSA	Key parameters L=1024, N=160 not approved

Service	*Role*	*Algorithm*	*Notes*
Parameter generation / Key generation / Shared secret computation	User	Diffie-Hellman	Prime bit sizes less than 2048 not approved
Shared secret computation	User	EC Diffie-Hellman	Curve P-192 not approved Brainpool curves (224, 256, 384, 512 bits)
Key generation / Digital signature generation	User	ECDSA	Curve P-192 not approved Brainpool curves (224, 256, 384, 512 bits) Curve25519 and ed448
Digital signature verification	User		Brainpool curves (224, 256, 384, 512 bits)
Symmetric Algorithms			
encryption/ decryption	User	ChaCha20	With use of Poly1305 algorithm
4768-001(CEX6A)			
Public Key Algorithms			
Digital signature generation	User	RSA	Key sizes less than 2048 not approved
Key Wrapping			Key sizes less than 2048 not approved
Shared secret computation	User	Diffie-Hellman	Key sizes less than 2048 not approved
Message Authentication Codes (MACs)			
Message Authentication	User	MD5	N/A

4 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 z14 mainframe computer.

CEX6A is the term used to refer to IBM 4768/Crypto Express6S hardware devices when configured on an IBM Z® server as hardware cryptographic accelerators.

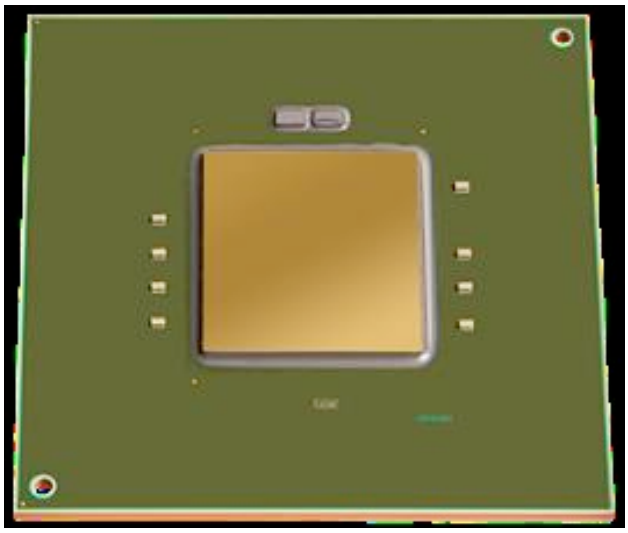
The CP Assist for Cryptographic Function (CPACF) (see Figure 5) is also a hardware device – part of the Co-Processor Unit (CoP). It offers the full complement of the Triple-DES algorithm, Advanced Encryption Standard (AES) algorithm and Secure Hash Algorithm (SHA), the AES-GCM algorithm, a true random-number generation function(TRNG/NDBRG), and SHA-512 deterministic random-number generation (DRNG/DBRG) function. Security Level 1 is satisfied by the device (CoP) being included within the physical boundary of the module and the device being made of commercial-grade components.

CPACF Physical Design: Each microprocessor (core) on the 8-core chip has its own dedicated 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.

Figure 3: IBM System z14 Mainframe Computer

Figure 4: Crypto Express6 Card

Figure 5: Processor Unit with CPACF COP



5 Operational Environment

5.1 Module Operation

The ICSF PKCS#11 module is intended to operate within z/OS Version 2 Release 4 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,* Chapter 2, Steps to customize SYS1.PARMLIB.
- ICSF must operate FIPS restricted for all applications requiring FIPS adherence.
- Applications requiring FIPS adherence must follow the recommendations found in SP 800-131A Revision 1.

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. When operating FIPS restricted, the module will not allow heritage (prior to SP 800-131A Revision 1) non-approved algorithms to be used. The module also offers non-approved but allowed RSA key establishment and exchange services even when operating FIPS restricted. Operating FIPS restricted automatically inhibits parameter combinations that are technically possible, but not permitted by heritage FIPS 140-2 restrictions.

Note that the module does not enforce the more recent restrictions introduced by SP 800-131A Revision 1, even when operating FIPS restricted. In some cases, it is not possible for ICSF to do the enforcement since the context of the request is not known. Therefore, all applications requiring FIPS adherence must explicitly follow the recommendations found in SP 800-131A Revision 1.

The ICSF PKCS #11 module (CSFINPV2), ICSF AP Service Interface (APSI), CPACF, and one optional Crypto Express card configured as an accelerator (CEX6A) represent 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 Crypto Express6 accelerator. No cryptographic operations are being performed on the data being provided by the ICSF PKCS #11 module until it arrives at the Crypto Express6 accelerator.

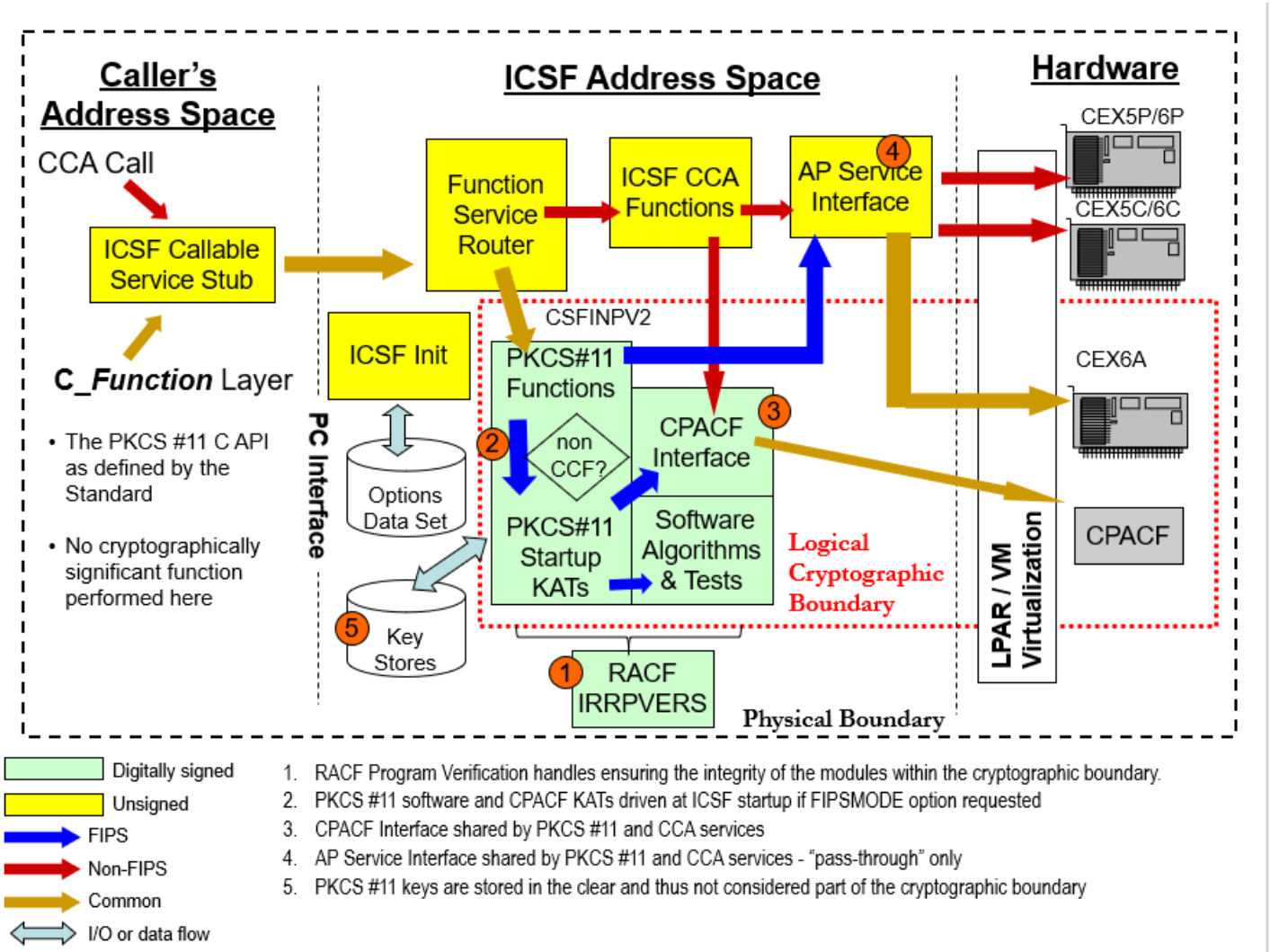
Conversely, the CPACF Interface does perform cryptographically relevant function and, thus, is physically part of the CSFINPV2 signed program object.

The RACF Signature Verification module (IRRPVERS) is shipped as part of the Security Server RACF FMID. IRRPVERS is bound by this module in order to validate the signature on CSFINPV2. It is not considered part of the cryptographic boundary of this module.

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 6, ICSF PKCS #11 cryptographic module software components are completely contained within the ICSF started task address space. The CEX6A and CPACF hardware components reside below the LPAR / VM virtualization layer. (Any IBM Crypto Express adapter configured with the Enterprise PKCS #11 (EP11) firmware, if installed, must be deactivated.) The CEX6C, if installed, is not used by the PKCS #11 cryptographic module when FIPS restrictions are being enforced. ICSF Callable Service stubs, instantiated in the caller's 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.

Figure 6: ICSF PKCS #11 Cryptographic Module



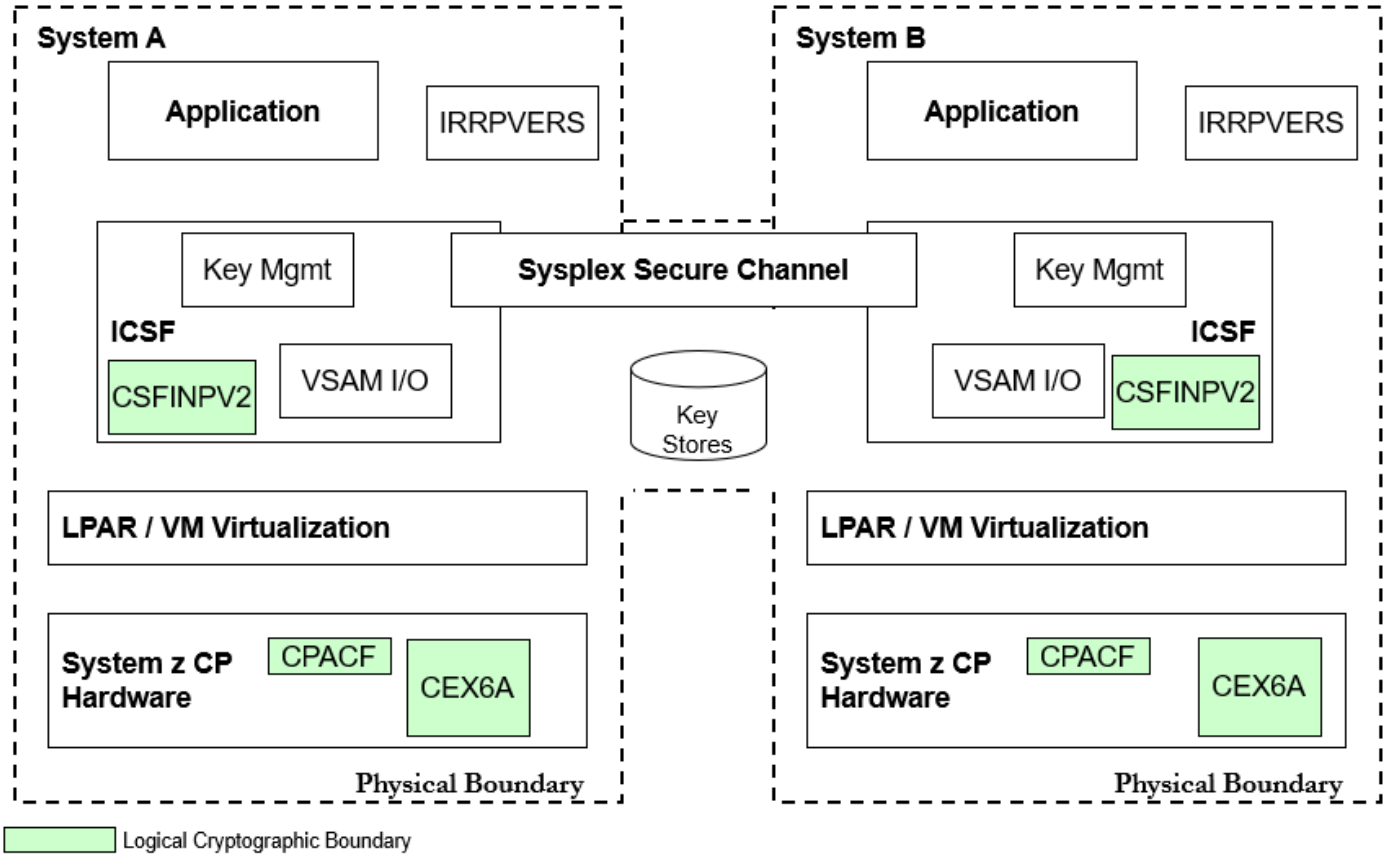
As shown in figure 7, ICSF PKCS #11 cryptographic module may be deployed in a high availability environment where ICSF and other applications may be in effect instantiated on multiple z/OS system instances configured in a "clustered" environment known as a parallel sysplex. A parallel sysplex enables 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.

Figure 7: ICSF PKCS #11 Cryptographic Module in a z/OS Sysplex Environment



6 Key Management

6.1 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. In particular, see

Table 8 and the footnotes of Table 6 for information on deprecated key sizes/usages.

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. In addition, persistent keys are stored in key store data sets (disk storage). The disk storage is outside the logical boundary.

6.2 Key Generation

Key generation uses random bytes produced by an approved RNG algorithm (specified in **NIST SP 800-90A**) which is known as Hash_DRBG (DRBG). The DRBG is hardware (CPACF) based and is a SHA-512 Hash DRBG with a security strength of 256 bits.

The DRBG is seeded by hardware based (CPACF) TRNG.

Note that in the case of the Vendor Affirmed configurations, a combination software and hardware based DRBG is used.

The Key Generation methods implemented in the module for Approved services in FIPS mode is compliant with Cryptographic Key Generation (CKG) for symmetric and asymmetric keys as per [SP800-133] (vendor affirmed). Symmetric key generation uses the DRBG directly to produce the key material.

DSA and ECDSA key generation is done according to FIPS Pub 186-4 [3]. RSA key generation implements only the FIPS Pub 186-4 key generation method. A non-compliant RSA key generation method is also present, which allows for the generation of RSA keys shorter than 1024 bits (FIPS Pub 186-4 does not permit generation of shorter keys), but the module does not permit the generation of such short keys when FIPS restricted. Diffie-Hellman key generation is similar to DSA key generation. EC Diffie-Hellman key generation is similar ECDSA key generation. For generating RSA, DSA and ECDSA keys the module implements asymmetric key generation services compliant with [FIPS186-4] and [SP800-90A]. A seed (i.e. the random value) used in asymmetric key generation is directly obtained from the [SP800-90A] DRBG.

6.3 Key Entry and Key Output

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 a wrapped (encrypted) format. Private keys may be wrapped using AES-GCM with 128, 192, or 256-bit AES keys, or any approved AES block chaining mode with HMAC or GMAC. Symmetric keys may be wrapped with RSA PKCS #1 v1.5, using any RSA key size ranging from 2048 to 4096 bits.

Keys may be unwrapped using AES-GCM with 128, 192, or 256-bit AES keys, or any AES approved block chaining mode with HMAC or GMAC, 3-key Triple-DES w/CBC PAD; 128, 192, 256-bit AES w/CBC PAD, or RSA PKCS #1 v1.5, using any RSA key size ranging from 1024 to 4096 bits.

Note: The unwrapping of keys using AES or Triple-DES is allowed according to IG D.9 (01/19/2018 version).

For FIPS 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.

6.4 Key Establishment

The module provides support for asymmetric key establishment methods as allowed by Annex D in the FIPS 140-2. The supported asymmetric key establishment methods are RSA Key Wrapping, Diffie-Hellman, and EC Diffie-Hellman shared secret computation.

When using Diffie-Hellman the module allows key lengths of 2048 bits which provides 112 bits of encryption strength. Use of a prime length less than 2048 bits is not allowed as per NIST SP 800-131A. Applications requiring FIPS adherence must not use prime lengths less than 2048 bits.

When using Elliptic Curve Diffie-Hellman the module allowed curves are P-224, P-256, P-384, and P-521 which provide between 112 and 256 bits of encryption strength. Use of the P-192 curve is not allowed as per NIST SP 800-131A for signature generation. Applications requiring FIPS adherence must not use curve P-192.

When using RSA Key Wrapping the allowed modulus lengths must be between 2048 and 4096 bits which provides between 112 and 150 bits of encryption strength. Use of a modulus length less than 2048 bits is not allowed as per NIST SP 800-131A. Applications requiring FIPS adherence must not use modulus lengths less than 2048 bits.

6.5 Key Protection

To enforce compliance with FIPS 140-2 key management requirements on the ICSF PKCS #11 module, code issuing calls must manage keys in a FIPS 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 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 role. It is the responsibility of application program developers to protect keys exported from the ICSF PKCS #11 module.

6.6 Key Destruction

ICSF PKCS #11 objects, when released on behalf of a caller are zeroized by ICSF. Local copies of all secret and private key object material made by ICSF are zeroized when they are no longer needed. In addition, z/OS ensures that the real storage frames that once backed the storage released are zeroed before the frames are reallocated to another caller. For persistent key objects stored in DASD key stores, the module's Crypto Officer may initiate the zeroization of those persistent key objects with the following procedure:

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.

6.7 Key/CSP Management Table

Table 9: Key/CSP Management Table

Key/CSP	Generation	Entry/output	Zeroization	Storage
AES key	CKM_AES_KEY_GEN (using the output of the SP800-90A DRBG)	Passed to and from the calling application	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF

				address space memory
Triple-DES key	CKM_DES3_KEY_GEN (using the output of a SP800-90A DRBG)	Passed to and from the calling application	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory
HMAC key	CKM_GENERIC_SECRET_KEY_GEN (using the output of a SP800-90A DRBG)	Passed to and from the calling application	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory
DRBG Internal State (C, V values)	NDRNG and derived internal state values based on SP800-90A DRBG	N/A	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory
RSA private key	CKM_RSA_PKCS_KEY_PAIR_GEN using FIPS 186-4	Passed to and from the calling application	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory
DSA private key	CKM_DSA_KEY_GEN using FIPS 186-4	Passed to and from the calling application	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory
ECDSA private key	CKM_EC_KEY_PAIR_GEN using FIPS 186-4	Passed to and from the calling application	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory
Diffie-Hellman private key	CKM_DH_KEY_GEN using FIPS 186-4	Passed to and from the calling application	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory
EC Diffie-Hellman private key	CKM_EC_KEY_PAIR_GEN using FIPS 186-4	Passed to and from the calling application	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory
Diffie-Hellman and EC Diffie-Hellman shared secret	computed using SP 800-56A	N/A	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory

Table 10: Key/CSP Management Table

Public Key	Generation	Entry/output	Zeroization	Storage
RSA public key	CKM_RSA_PKCS_KEY_PAIR_GEN using FIPS 186-4	Passed to and from the calling application	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory
DSA public key	CKM_DSA_KEY_GEN using FIPS 186-4	Passed to and from the calling application	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory
ECDSA public key	CKM_EC_KEY_PAIR_GEN using FIPS 186-4	Passed to and from the calling application	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory
Diffie-Hellman public key	CKM_DH_KEY_GEN using FIPS 186-4	Passed to and from the calling application	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory
EC Diffie-Hellman public key	CKM_EC_KEY_PAIR_GEN using FIPS 186-4	Passed to and from the calling application	Zeroized when the cipher context is released	Persistent keys are stored in a key store (outside the module's boundary) ICSF address space memory

6.8 APIs

The following services (APIs) in

Table 11 can be executed by the User role. 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 shared secret computation, key parameter and key generation
- EC Diffie-Hellman shared secret computation and key generation
- DRBG.

Table 11: ICSF PKCS #11 Module Services (APIs)

Note: API marked as strikethrough as disabled when the module is configured according to the Crypto Officer and User guidance specified in section 9.2

Note: The Approved AES key wrapping shall be compliant with SP800-38F

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 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_DES3_KEY_GEN CKM_AES_KEY_GEN CKM_DSA_PARAMETER_GEN CKM_DH_PKCS_PARAMETER_GEN CKM_GENERIC_SECRET_KEY_GEN 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_MD5 CKM_SHA_1 CKM_SHA224 CKM_SHA256 CKM_SHA384 CKM_SHA512 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_SHA1_RSA_PKCS_PSS CKM_SHA224_RSA_PKCS_PSS CKM_SHA256_RSA_PKCS_PSS CKM_SHA384_RSA_PKCS_PSS CKM_SHA512_RSA_PKCS_PSS 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 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 • CKM_ECDSA	C_Sign C_SignFinal C_Decrypt
CSFPPS2	PKCS #11 Private key structure sign	Sign data using an RSA private key structure using PKCS #1 1.5 formatting.	Direct application call
CSFPPD2	PKCS #11 Private key structure decrypt	Decrypt data using an RSA private key structure using PKCS #1 1.5 formatting	Direct application call
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 • CKM_ECDSA	C_Verify C_VerifyFinal C_Encrypt

CSFPPV2	PKCS #11 Public key structure verify	Verify data using an RSA public key structure using PKCS #1 1.5 formatting.	Direct application call
CSFPPE2	PKCS #11 Public key structure encrypt	Encrypt data using an RSA public key structure using PKCS #1 1.5 formatting.	Direct application call
CSFPSKD	PKCS #11 Secret key decrypt	Decipher data using a clear symmetric key Supports mechanisms: CKM_DES3_ECB CKM_DES3_CBC CKM_DES3_CBC_PAD CKM_AES_CBC CKM_AES_ECB CKM_AES_CBC_PAD CKM_AES_GCM CKM_AES_CTS -(Plus CTR rule)	C_Decrypt C_DecryptUpdate C_DecryptFinal
CSFPSKE	PKCS #11 Secret key encrypt	Encipher data using a clear symmetric key Supports mechanisms: CKM_DES3_ECB CKM_DES3_CBC CKM_DES3_CBC_PAD CKM_AES_CBC CKM_AES_ECB CKM_AES_CBC_PAD CKM_AES_GCM CKM_AES_CTS -(Plus CTR rule)	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_DES3_CBC_PAD CKM_AES_CBC_PAD CKM_AES_GCM	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	C_Verify C_VerifyUpdate C_VerifyFinal
CSFPWPK	PKCS #11 Wrap key	Wrap a key with another key Supports mechanisms: CKM_RSA_PKCS CKM_AES_CBC_PAD CKM_AES_GCM	C_WrapKey

7 EMI/EMC

Systems utilizing the module’s services have their overall EMI/EMC ratings determined by the host system, which includes the CPACF. The validation environments meet the requirements of 47 CFR FCC PART 15, Subpart B, Class A (Business use).

EMI/EMC requirements for the Crypto Express6 cards are met by the card's FCC Class B rating.

8 Self-Tests

8.1 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 the DRBG is used and when asymmetric keys are generated. These tests include a continuous random number generator test and pair-wise consistency tests of the generated DSA, ECDSA or RSA keys.

8.2 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 either terminates or is rendered unusable (all cryptographic services return an error return code).

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 starts with bound module IRRPVERS verifying its own digital signature. Once verified, IRRPVERS verifies the digital signature of CSFINPV2.

Algorithm known answer tests are performed by the module. The tests are performed prior to any cryptographic algorithms being executed. All output from the module is inhibited until the self-tests have completed.

The module tests the following cryptographic algorithms:

Table 12: Self-Tests

ICSF Component	Algorithm	Method
CPACF	AES	KAT (encrypt/decrypt tested separately for CBC, ECB, CTR and GMAC)
	Triple-DES	KAT (encrypt/decrypt tested separately for CBC and ECB)
	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	KAT
	Hash_DRBG	KAT
		Health tests (per section 11.3 of SP 800-90A)
Software	AES	KAT (encrypt/decrypt tested separately) for CBC and GCM
	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	KAT
	HMAC	KAT (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)
	RSA	KAT (sign/verify and encrypt/decrypt) with 2048-bit key
	DSA	PCT (sign/verify) with 2048-bit key
	ECDSA	PCT (sign/verify) with P-256
	Diffie-Hellman	KAT (shared secret computation) with 2048-bit key
	EC Diffie-Hellman	KAT (shared secret computation) with P-256
	Hash_DRBG	KAT
		Health tests (per section 11.3 of SP 800-90A)
CEX6	RSA	KAT (sign/verify and encrypt/decrypt) with 2048-bit key
	Diffie-Hellman	KAT (shared secret computation) with 2048-bit key

Self-tests are performed in logical order, verifying module integrity incrementally:

- Integrity test on module, using RSA/SHA-256 (Performed by IRRPVERS when the module is loaded)
- Known-answer tests on all Approved algorithms except DSA and ECDSA
- Pairwise consistency test for DSA and ECDSA.

Once the startup self-tests are complete (and successful), the module will issue console message CSFM015I.

8.3 Conditional Self-Testing - 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.

8.4 On-Demand Self-Tests

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.

8.5 Crypto Express6 (CEX6) Cards

The IBM Crypto Express6 features 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 the automatic self-tests, other 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 Express6 conditional data tests that are applicable to their use in this security policy are continuous integrity checks on modular math arithmetic (including RSA and DSA operations), implemented in hardware.

The Crypto Express6 cards will be left offline if any of the startup self-tests fail. The ICSF PKCS #11 module is unaffected by this action.

To execute the self-tests on demand, the Crypto Express6 cards required a reboot from the hardware management console.

9 Crypto Officer and User Guidance

9.1 Installation and Invocation

ICSF level HCR77D0 is installed as part of the z/OS Version 2 Release 4 ServerPac. In addition, it is also 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 OA58593 and is bound to the IRRPVERS module.

After installation, ICSF must be started to make the ICSF PKCS #11 module available. The Crypto Officer must specify the desired FIPSMODE option in the ICSF options data set which is used at ICSF start up or for the SETICSF OPTIONS,REFRESH command:

FIPSMODE can be changed to YES or COMPAT only if the ICSF PKCS#11 Startup Self-Tests passed and the Conditional Self-Testing has passed.

FIPSMODE(YES,FAIL(YES|NO))

- FIPS standard mode – Provides full FIPS support. All applications must adhere to the FIPS restrictions that were in effect circa 2009 (prior to SP 800-131A Revision 1 [6]). These are henceforth known as the heritage FIPS restrictions. All heritage FIPS restrictions are enforced. Any request for a disallowed heritage cryptographic function is denied.

FIPSMODE(COMPAT,FAIL(YES|NO))

- FIPS compatibility mode – Provides FIPS support for select applications. Applications that are not “FIPS Exempt” must adhere to heritage FIPS restrictions. Any request for a disallowed heritage cryptographic function using a FIPS only key or requested by a non-FIPS Exempt application is denied.

FIPSMODE(NO,FAIL(YES|NO))

- FIPS no enforcement mode, also known as FIPS On-Demand Mode – Provides FIPS support for applications that explicitly request FIPS adherence. Any request for a disallowed heritage cryptographic function by an application explicitly requesting FIPS adherence or using a FIPS only key is denied. This is the default mode.

These options are fully described in the programming documentation.

***Note**: Whenever ICSF is enforcing the heritage (prior to SP 800-131A Revision 1) FIPS restrictions, it is considered to be running “FIPS restricted.” FIPS restrictions prior to SP 800-131A Revision 1 are enforced by ICSF. Nevertheless, FIPS restricted is not the same as FIPS-Approved mode of operation. Since 2009, the CMVP has disallowed more algorithms and key lengths through SP 800-131A and its subsequent two revisions. The FIPS-Approved mode of operation is defined by Service Tables 6 and 7 in section 5.2 of this Security Policy (known as “enforced by policy). Invoking a FIPS-Approved or Allowed service will make the module enter the FIPS-Approved mode of operation, whereas invoking a non-Approved and non-Allowed service will make the module enter the non-FIPS Approved mode of operation. This is irrespective of which FIPSMODE was selected for the module.*

The cryptographic module is invoked via the APIs, as documented in the *z/OS Cryptographic Services ICSF Application Programmer's Guide* and *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications.*

The CPACF Enablement Feature 3863 must be installed prior to starting ICSF. This feature code may be ordered from IBM then downloaded through RETAIN and installed using the Hardware Management Console (HMC).

The hardware accelerator(CEX6A) is optionally installed and configured by an IBM customer engineer (CE) in accordance with the card's manual and installation procedures. The customer uses the HMC to assign it to the logical partition where ICSF is residing. The mode of operation as an accelerator (vs a coprocessor) is configured using the HMC.

Note, if a CEX6A is to be used, there must be only one CEX6A assigned to the partition where ICSF resides. Furthermore, the single accelerator in use must be configured online prior to starting ICSF and it must not be reconfigured while ICSF is running.

Any firmware loaded into this module that is not shown on the module's certificate is out of scope of this validation, and requires a separate FIPS 140-2 validation.

9.2 Module Configuration for FIPS 140-2 Compliance

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

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

  For proper operations, the Crypto Officer 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 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. Applications must also comply with the key size and algorithm requirements specified in the latest revision of NIST Special Publication 800-131A Revision 1.
- Operating with an active Crypto Express coprocessor configured as a secure key PKCS #11 coprocessor (CEX6P for example) is not an approved platform configuration. The user or Crypto Officer must ensure that there are no active secure key PKCS #11 coprocessor configured.
- 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.
- The module complies with scenario 3 of IG A.5. If any application will use PKCS #11 objects for AES Galois/Counter Mode (GCM) encryption or GMAC generation, and will have ICSF generate the initialization vectors, then you need to set ECVTSPLX (sysplex name) or CVTSNAME (system name if not in a sysplex) to a unique value. A unique value requires that the first four bytes of the sysplex name or system name are unique.

  In case the module's power is lost and then restored, a new key and IV for use with the AES GCM encryption/decryption shall be established.
- According to IG A.13, the same Triple-DES key shall not be used to encrypt more than 2^{16} 64-bit blocks of data.

- **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 CRYPTO OFFICER**
    - The system administrative personnel are not careless, willfully negligent, or hostile, and will follow and abide by the instructions provided by the Crypto Officer 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.

10 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.

11 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 8 illustrates the IBM z14 with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and CPACF System Driver Level 32L configuration (Base GPC).

Figure 8: Validated Configuration with CPACF only

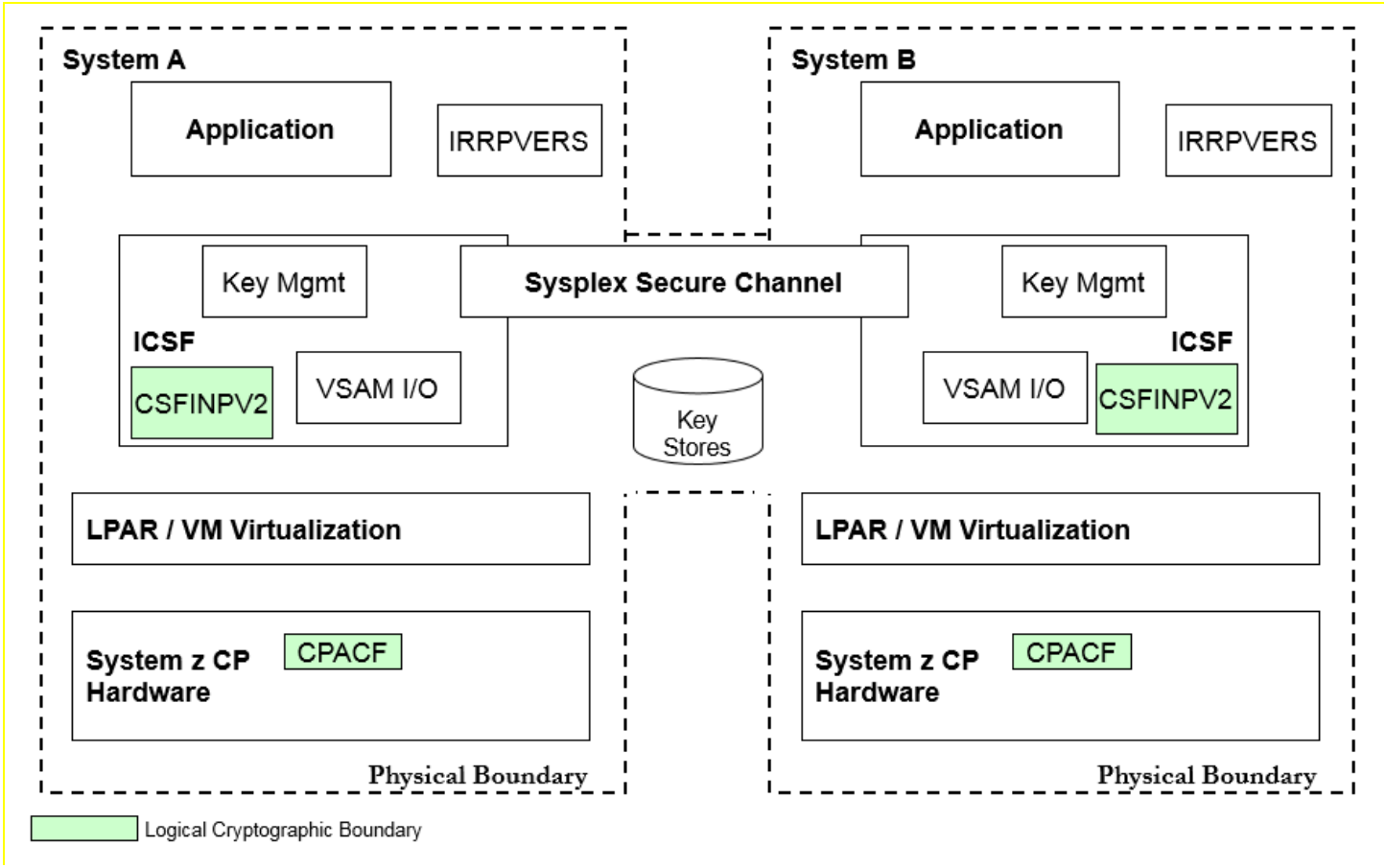
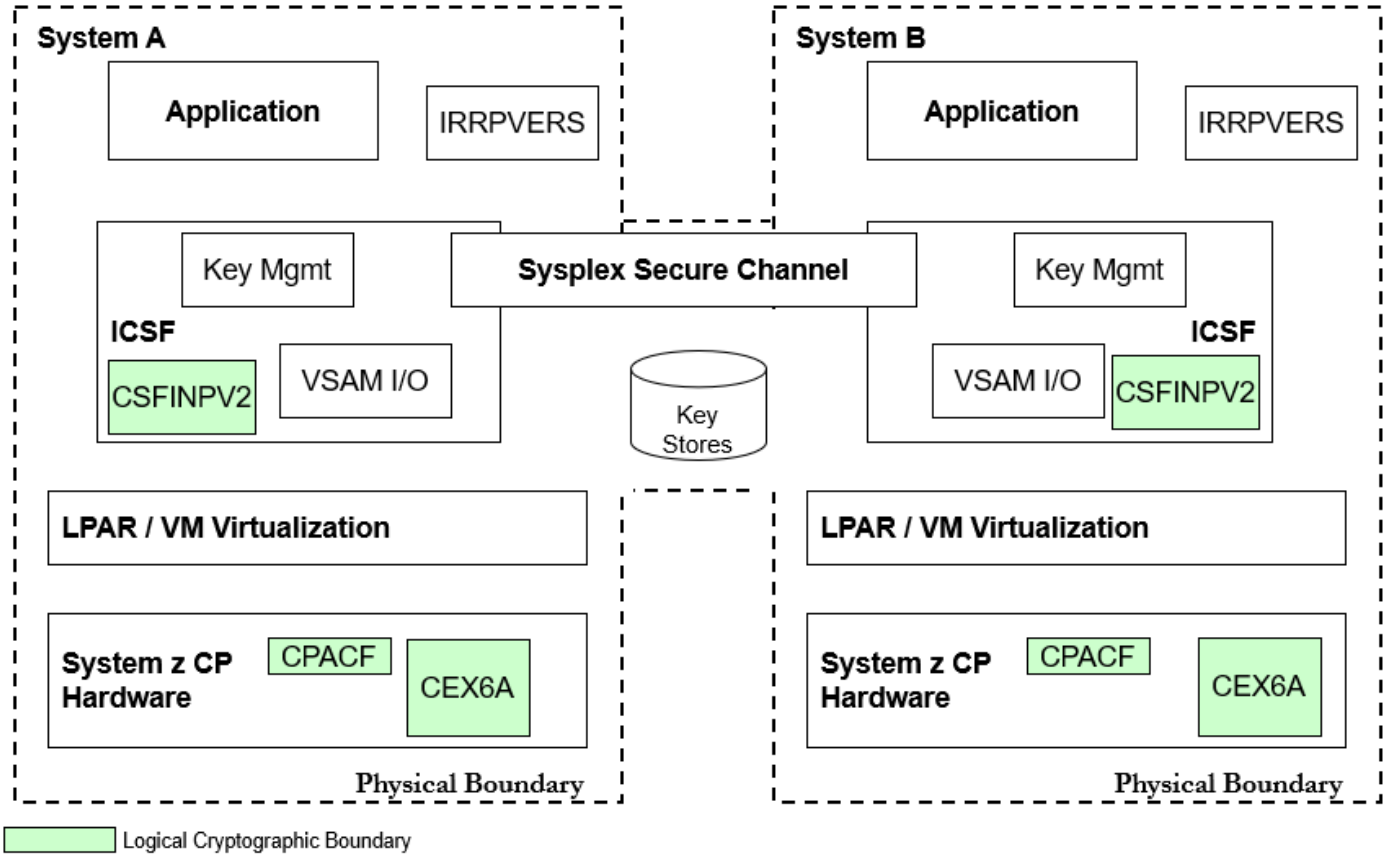


Figure 9 illustrates the IBM z14 with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and CPACF System Driver Level 32L and Crypto Express cards (Accelerator CEX6A) configuration.

Figure 9: Validated Configuration with CPACF and CEX6A



12 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
CCA	IBM's Common Cryptographic Architecture, a standard for performing cryptographic functions.
CEX5S	Crypto Express5S, the IBM Z mainframe server name for the IBM 4767 PCIe Cryptographic Coprocessor 5 hardware security module (HSM).
CEX5A	Crypto Express5 Accelerator, a CEX5S configured as an accelerator.
CEX5C	Crypto Express5 Coprocessor, a CEX5S configured as coprocessor supporting the IBM Common Cryptographic Architecture.
CEX5P	Crypto Express5 Enterprise PKCS #11 secure key Coprocessor, a CEX5S configured as a coprocessor supporting the PKCS #11 standard.
CEX6S	Crypto Express6S, the IBM Z mainframe server name for the IBM 4768 PCIe Cryptographic Coprocessor 6 hardware security module (HSM).
CEX6A	Crypto Express6 Accelerator, a CEX6S configured as an accelerator.
CEX6C	Crypto Express6 Coprocessor, a CEX6S configured as coprocessor supporting the IBM Common Cryptographic Architecture.
CEX6P	Crypto Express6 Enterprise PKCS #11 secure key Coprocessor, a CEX6S configured as a coprocessor supporting the PKCS #11 standard.
CP	Central Processor, aka CPU.
CPACF	CP Assist for Cryptographic Function, clear key on-chip accelerator integrated into mainframe processors. CPACF functionality is restricted to symmetric, hashing and random number generation 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.
DRBG	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]
HMC	The Hardware Management Console is a feature on IBM z Systems that provides an interface to control and monitor the status of the IBM z system.
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]
RETAIN	IBM database system shared by IBM and its customers
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/NDRNG	True Random Number Generator (or Non-Deterministic 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 DRBG.

13 References

[1] z/OS V2R4 elements and features PDF files - Cryptographic Services
https://www.ibm.com/support/knowledgecenter/SSLTBW_2.4.0/com.ibm.zos.v2r4.e0zc100/crypt.htm

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

[3] National Institute of Standards and Technology, Federal Information Processing Standards, Digital Signature Standard (FIPS 186-4), 2013
[4] ABCs of z/OS System Programming Volume 1 (SG24-6981-01)

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

[6] National Institute of Standards and Technology, Special Publication 800-131A Revision 2, Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths, March 2019

[7] PKCS #11 Cryptographic Token Interface Base Specification Version 2.40. http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.pdf

14 Trademarks

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

- IBM
- RACF
- z13
- z14
- IBM Z
- z/OS.