IBM® z/OS® Version 1 Release 12 System SSL Cryptographic Module

FIPS 140-2

Non-Proprietary Security Policy

Policy Version 1.01

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

IBM Research
Zurich Research Laboratory

August 24, 2011

Table of Contents

1 Scope of Document

This document describes the services that the z/OS System SSL library (“System SSL” 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 System SSL library in its FIPS 140-2 configuration consists of a set of loadable modules. A portion of the System SSL shared library binaries consists of both 31- and 64-bit versions. The deployed version consists of the following modules:

Table 1 System SSL Library Modules

31-bit	64-bit	Auxiliary
GSKSSL	GSKSSL64	GSKSCTSS
GSKS31F	GSKS64F	GSKSUS31
GSKCMS31	GSKCMS64	GSKSUS64
GSKC31F	GSKC64F	GSKMSGXT
GSKSRBRD		GSKWTR
GSKSRBWT		GSKSCTFT
IRRPVERS		GSKS31
GSKSRVR		GSKS64
GSKKYMAN		GSKC31
		GSKC64
		GSKRACF
		IRRVERLD
		Header Files
		Side Decks
		Message Catalogs
		Sample Client/Server Application

The z/OS System SSL 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 either provide functionality that is not cryptographically relevant (sample applications) or are not utilized while operating in FIPS 140-2 mode. The files consist of header, side decks, message catalogs, sample client/server applications, sample configuration scripts and component trace.

The z/OS System SSL logical and physical boundaries are described in Figures 1 and 2 in the Operational Environment Section.

Note: Throughout this document,

- the Crypto Express3 cards will also be referenced using the terms CEX3, CEX3C, CEX3A, and 4765-001
- the CP Assist for Cryptographic Functions will also be referenced using the terms CP Assist and CPACF.

2 Cryptographic Module Specification

The z/OS System SSL 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 System SSL module running in configurations backed by hardware cryptography. The System SSL module consists of software-based cryptographic algorithms, as well as symmetric and hashing algorithms provided by the CP Assist for Cryptographic Function (CPACF), Elliptic Curve Cryptography provided by ICSF and RSA Hardware clear key modular math cryptography provided through the Crypto Express cards (CEX3C and CEX3A). The hardware support is accessed through the z/OS Integrated Cryptographic Services Facility (ICSF) acting as a "pipe" between System SSL and the cryptographic cards.

Table 2 System SSL Module Components

Type/Name	Version
Software Components System SSL DLLs (APIs) System SSL Executables (GSKKYMAN, GSKSRVR) RACF IRRPVERS ICSF Secure Channel	z/OS Version 1 Release 12 with System SSL level HCPT3C0/JCPT3C1 with APAR OA34156, RACF level HRF7770 and ICSF level HCR7770 with APAR OA34205
Hardware Components CPACF 4765-001	 CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 with System Driver Level 86E Firmware – e1ced7a0 Hardware – 45D6048

System SSL validation was performed using the z/OS Version 1 Release 12 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 Crypto Express3 card (Coprocessor (CEX3C))

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

4. IBM zEnterprise™ 196 (z196) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and Crypto Express3 (Coprocessor (CEX3C) and Accelerator (CEX3A)) cards.

The module running on the above platforms was validated as meeting all **FIPS Pub 140-2** Level 1 security requirements. The z/OS System SSL module is packaged as a set of DLLs and executables which contains all the code for the module. The library is accompanied by its primary header files, gskcms.h and gskssl.h. (Other support files, such as auxiliary headers or link files are included in the distribution.)

See Section 13, Cryptographic Module Configuration Diagrams, for more information about the validated platforms.

In addition to the configurations tested by the laboratory, vendor-affirmed testing was performed using z/OS Version 1 Release 12 on the following platforms

1. IBM System z10® Enterprise Class (z10 EC) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 (Base GPC)
2. IBM System z10® Enterprise Class (z10 EC) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and Crypto Express3 card (Coprocessor (CEX3C))
3. IBM System z10® Enterprise Class (z10 EC) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and Crypto Express3 card (Accelerator (CEX3A))
4. IBM System z10® Enterprise Class (z10 EC) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and Crypto Express3 (Coprocessor (CEX3C) and Accelerator (CEX3A)) cards.
5. IBM System z10® Business Class (z10 BC) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 (Base GPC)

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

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	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

4 Ports and Interfaces

As a multi-chip standalone module, the System SSL physical interfaces are the boundaries of the host running System SSL library code. The underlying logical interfaces of the module are the C language Application Programming Interfaces (APIs) documented in the System SSL reference manual, `z/OS Cryptographic Services Secure Sockets Layer Programming' [1] 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 dedicated parameters of the public gsk_fips_state_set API and the GSK_FIPS_STATE environment variable.

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

Status output is provided in return codes and through messages. Documentation for each API lists possible return codes. A complete list of all return codes returned by the C language APIs within the System SSL module is provided in the external header files. Messages are documented in the System SSL reference manual, `z/OS Cryptographic Services Secure Sockets Layer Programming' and RACF manual, '*z/OS Security Server RACF Messages and Codes'*.

Cryptographic bypass capability is not supported by System SSL.

Module Status The System SSL library communicates any error status synchronously through the use of its documented return codes. It is the responsibility of the calling application to handle exceptional conditions in a FIPS 140-2 appropriate manner. For a textual description of the C language return code, applications can use the gsk_strerror API.

System SSL is optimized for library use and does not contain any terminating assertions or exceptions. Any internal error detected by System SSL and not induced by user data will be reflected back to the application with an appropriate return code. The calling application must examine the return code 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 codes and error conditions are fully documented in the product’s programming documentation

5 Roles, Services and Authentication

5.1 Roles

The module supports two roles: a cryptographic officer (Officer) role and a User role (Table 4). The module does not support user identification or authentication that would allow the module to distinguish between the two supported roles. Each of the roles is authenticated through the operating system prior to using any system services.

The Officer role is a purely administrative role that does not involve the use of cryptographic services. The role is not explicitly authenticated but assumed implicitly on implementation of the module’s installation and usage sections defined in the security rules section.

The User role has access to all of the module’s services. The role is not explicitly authenticated, but assumed implicitly on access of any of the non-Officer services. An operator is implicitly in the User or Officer role based upon the service(s) chosen. If any of the User-specific services are called, then the operator is in the User role; otherwise the operator is in the Officer role.

Table 4 Roles and Authentication Mechanisms

Role	Type of Authentication	Authentication Data	Strength of Mechanism
Officer	None(Automatic)	None	N/A
User	None(Automatic)	None	N/A

5.2 Services

The module provides queries and commands (Tables 7 and 8). Queries return status of commands or command groups; commands exercise cryptographic functions. Officers perform queries; Users may perform both queries and commands. 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 7.

Services are accessed through documented API interfaces from the calling application.

Certificate management services (CMS) perform both non-cryptographic and cryptographic PKI management activities, as well as general cryptographic operations, such as signature verification. Functions in this group parse and categorize X.509 certificates and transport certificates, and also handle standard encodings (such as PKCS#7). Cryptographic operations, such as signature verification, are delegated to lower-level crypto core functions.

SSL protocol implementation is split into infrastructure and protocol functions. Lower-level functions implement SSL message formatting and other primitives. SSL protocol operations extend SSL primitives with handshake state machines, session caching, and attribute parsing to provide a full SSL/TLS implementation. Both System SSL layers use cryptographic cores indirectly. SSL 3.0 functionality is disallowed in FIPS 140-2 mode: all other compliance checks are implemented at lower levels. In FIPS 140-2 mode, cipher suites are restricted to those built with approved algorithms only.

Format conversions, labeled as “other operations", are other non-cryptographic commands that change the representation of data. Format converters read and write, among others, the following formats:

- Various protocols based on ASN.1/BER encoded data (PKI-related and similar standard formats)
- Conversions between industry-standard object identifiers and internal symbolic constants (mainly intrinsic, not externalized).

Protocol-level format conversions generally package data without modification, treating output or input of lower-level crypto primitives as opaque data. The purpose of these conversions is to bridge protocols with predefined formats with cryptographic primitives, which are oriented around raw byte streams or blocks, but generally not standard encapsulation methods:

- Base-64 encoding (“ASCII armor"), generating and reading printable representation of binary data, for example, encountered in certificates
- Conversions between ASCII and non-ASCII data (such as EBCDIC), non-cryptographic but potentially modifying security-relevant data.

Format conversion services do not provide cryptographic functionality, but may use other services if the transport mechanism requires them. As an example, if signed data is represented as a standard ASN.1 structure, it implicitly uses one of the sign calls. Similarly, certificate management or processing of PKCS#7 data may involve signature verifications, for example.

For a list of API services available in the System SSL module, see Table 8.

`

Table 5 Services and Access

Service	**Notes**	**Modes**	**Approved If yes, Cert #**	**CSPs and Access**	
Software					
Symmetric Algorithms					
AES	128 or 256 bit keys (FIPS 197)	CBC	Yes Cert #1702 #1703	AES Symmetric key	Read/ Write
Triple DES	168 bit keys	CBC	Yes Cert #1093 #1094	Triple DES Symmetric key	Read/ Write
Public Key Algorithms					
DSA Key/Parameter Generation	1024 bit modulus (FIPS 186-2 key size)	N/A	Yes Cert #526 #527	DSA public and private key	Write
DSA Sign	1024 bit modulus	N/A	Yes Cert #526 #527	DSA private key	Read
DSA Verify	1024 bit modulus	N/A	Yes Cert #526 #527	DSA public key	Read
RSA Key Generation	ANSI X9.31 (1024 to 4096 bits)	N/A	Yes Cert #831 #832	RSA public and private key	Write
RSA Sign	PKCS#1 (1024 to 4096 bits)	N/A	Yes Cert #831 #832	RSA private key	Read
RSA Verify	PKCS#1 (1024 to 4096 bits)	N/A	Yes Cert #831 #832	RSA public key	Read
RSA Verify for module integrity	PKCS #1 using SHA-256 (IRRPVERS)	N/A	Yes Cert #846	RSA public key	Read
RSA Encrypt/Decrypt	PKCS#1 (1024 to 4096 bits)	N/A	No	RSA public and private key	Read
Diffie-Hellman (DH)	2048 bits modules	Key Agreement/ Generation	No	DH public and private key	Read/ Write
Hash Functions					
SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	FIPS 180-3	N/A	Yes Cert #1485 #1486	None	N/A
Message Authentication Codes (MACs)					
HMAC-SHA1	FIPS 198, 198a	N/A	Yes Cert #986 #987	HMAC-SHA-1 key	Write
HMAC-MD5	N/A	N/A	No	HMAC-MD5 key	Write

Service	Notes	Modes	Approved If yes, Cert #	CSPs and Access	
Random Number Generation					
RNG	FIPS 186-2, ANSI X9.31	N/A	Yes Cert #901 #902	Seed, Seed Key	Write
CP Assist for Cryptographic functions					
Symmetric Algorithms					
AES	128 or 256 bit keys (FIPS 197)	CBC	Yes Cert #1713	AES Symmetric key	Read/ Write
Triple DES	168 bit keys	CBC	Yes Cert #1103	Triple DES Symmetric key	Read/ Write
Hash Functions					
SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	FIPS 180-3 SHA-2 algorithms	N/A	Yes Cert #1497	None	N/A
4765-001 (Crypto Express3) – hybrid					
Public Key Algorithms					
RSA Verify	PKCS#1 (1024 to 4096 bits)	N/A	Yes Cert #844 #845	RSA public key	Read
RSA Encrypt/Decrypt	PKCS#1 (1024 to 4096 bits)	N/A	No	RSA public and private key	Read
Other functions					

Service	Notes	Approved
ECDSA Sign	NIST P-192 – P-521	Non-Compliant
ECDSA Verify	NIST P-192 – P-521	Non-Compliant

Table 6 Additional algorithms within cryptographic boundary in Non-FIPS mode

Service	Notes
DES encryption/decryption	Software
RC2 encryption/decryption	Software
ArcFour encryption/decryption	Software
MD2	Software
MD5	Software
DES	CPACF

Table 7 Queries

Service	Notes	Roles	
Module Status		**Officer**	**User**
Query mode	Check if module is in FIPS 140-2 mode (gsk_fips_state_query)	Yes	Yes
Integrity Checks			
Power-up Tests	Automatic before first use Includes gsk_perform_kat	(yes)	No
Self-Tests	gsk_perform_kat, IRRPVERS self-test requires system IPL	Yes	Yes
Operational Correctness Checks			
RNG Tests	Continuously performed (automatic)	Yes	Yes
Pair-wise consistency	Continuously performed (automatic)	Yes	Yes

6 Operational Environment

Installation and Invocation

System SSL level HCPT3C0 and JCPT3C1, RACF level HRF7770 and ICSF HCR7770 are installed as part of the z/OS Version 1 Release 12 ServerPac using the “Installing Your Order” documentation provided with the ServerPac (prepackaged tailored z/OS installation including z/OS System SSL, ICSF and RACF). The tested version of System SSL requires the installation of service provided through System SSL APAR OA34156 and ICSF APAR OA34205.

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

The module is accessed from C/C++ language programs through the inclusion of header file gskssl.h and/or gskcms.h, or through the R_PgmSignVer callable service. The module is also accessed through the invocation of the gskkyman certificate utility and GSKSRVR started task.

The hardware accelerator (CEX3A) and hardware coprocessor (CEX3C) are installed and configured by an IBM customer engineer (CE). The hardware cards are selected and invoked through ICSF.

Module Operation

The System SSL 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 library 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 System SSL can access certain platform-specific functionality, which is not represented in higher-level languages, System SSL uses a mixture of high-level and platform-specific native code.

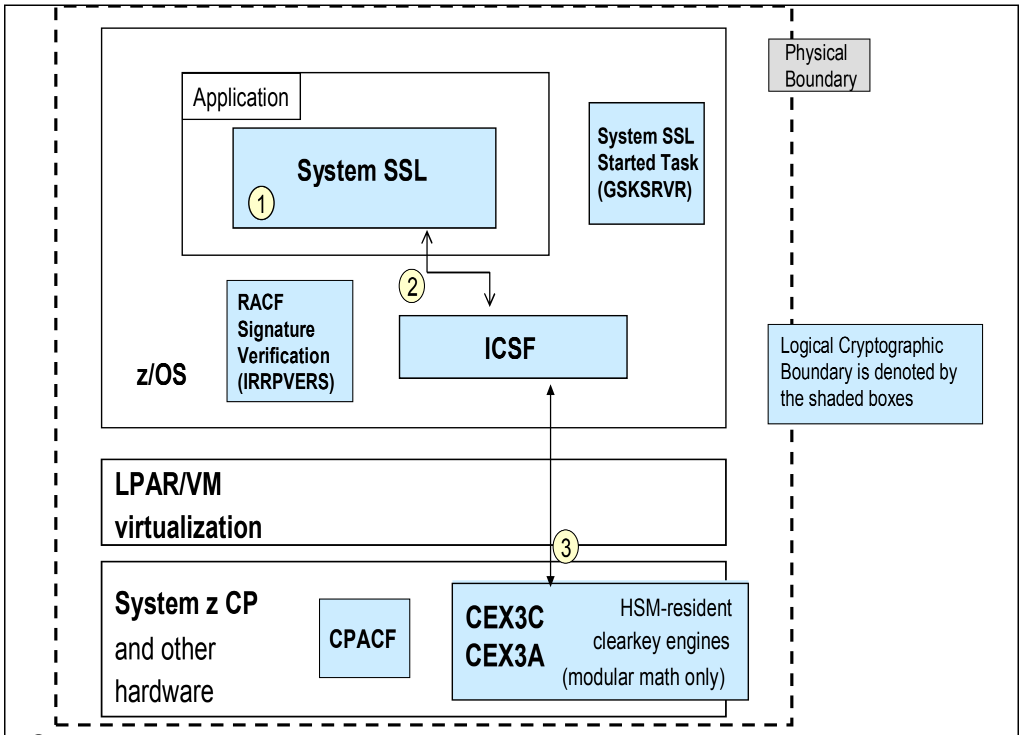
Using z/OS System SSL in a FIPS 140-2 approved manner assumes that the following defined criteria are followed:

- 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 application using the module services must consist of one or more processes in which each process is utilizing a separate copy of the executable code.
- The application designer must be sure that the application is designed correctly and does not corrupt the storage in the address space where the instance of System SSL is loaded.
- The unauthorized reading, writing or modification of either the GSKSRVR started task or application address space which contains the System SSL instance is not allowed.
- An instance of the System SSL Library DLLs must be accessed only by a single process (address space). This means that each process has it own instance of the System SSL Library DLLs.
- The System SSL setup procedures documented in the programming documentation must be followed and setup done correctly.
- The System SSL module must be initialized to execute in the FIPS 140-2 mode of operation. This is accomplished through the gsk_fips_state_set API.
- The CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 must be installed and enabled.
- The Crypto Express3 cards are installed and operated by properly trained personnel in accordance with the card's manual and installation procedures.

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 in addition to RSA/Diffie-Hellman for key establishment and exchange. The FIPS 140-2 configuration automatically inhibits parameter combinations that are technically possible, but not permitted in FIPS 140-2 mode.

The z/OS System SSL Application Programming Interfaces (APIs), RACF Signature Verification callable service (R_PgmSignVer), gskkyman certificate utility, GSKSRVR started task, ICSF, Secure Channel and CEX3C and CEX3A 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.

As shown in Figure 1, System SSL Cryptographic Module, the cryptographic module's DLLs are instantiated within an application's address space. Each application or operating system component that utilizes the System SSL cryptographic module will create a new instance of the z/OS System SSL DLLs. The System SSL Started task (GSKSRVR) and ICSF, which serves as "pass through" to the Crypto Express3 coprocessor card, are shared by instances of the System SSL DLLs loaded in the application address space(s) that are running in the z/OS Operating System environment. The RACF Signature Verification (IRRPVERS) module performs the initial integrity power-up tests.


Physical Boundary
Application
System SSL
1
System SSL Started Task (GSKSRVR)
2
RACF Signature Verification (IRRPVERS)
ICSF
z/OS
Logical Cryptographic Boundary is denoted by the shaded boxes
LPAR/VM virtualization
3
System z CP and other hardware
CPACF
CEX3C CEX3A
HSM-resident clearkey engines (modular math only)

1 The System SSL DLLs and SSL Started Task are considered within the cryptographic boundary. The System SSL DLLs may issue the System z CPACF machine instructions to perform symmetric encryption and hashing cryptographic functions that are provided by these machine instructions.

2 System SSL Calls ICSF via ICSF callable services for ECC Digital Signature Generate and Verification and for accelerated modular math RSA functions. System SSL invokes the ICSF RSA services with a clear RSA key and data to perform the following functions:

- Signature Verification
- Data Encryption and Decryption using the specified clear RSA Key

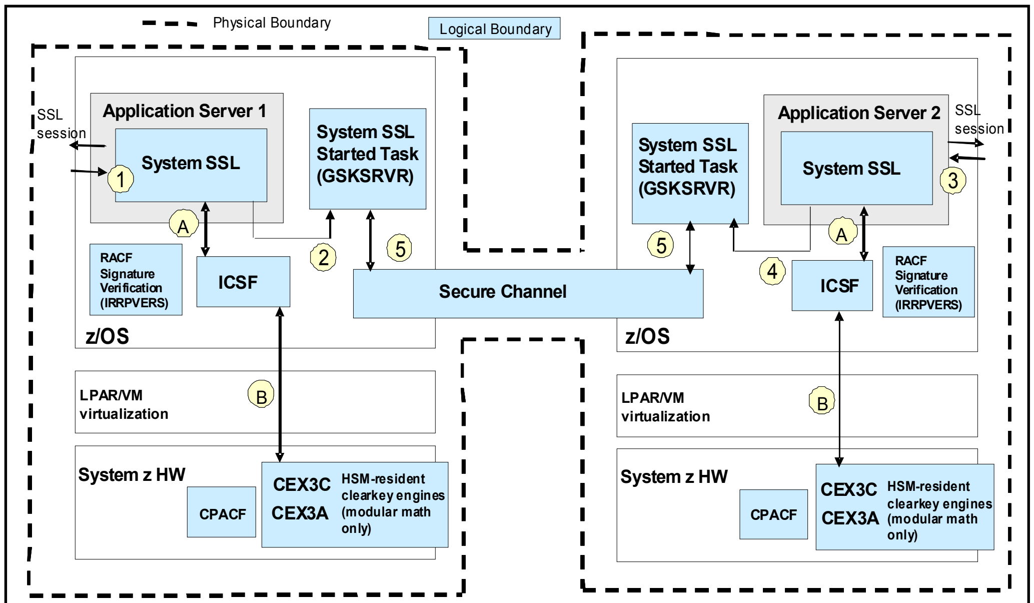
3 ICSF acts as a "pass-thru" routing the clear RSA key and data to the CEX3C or CEX3A which utilizes modular math hardware acceleration functions provided by the PCI card hardware.

Note: RACF Program Verification handles ensuring the integrity of the modules within the cryptographic boundary.

Figure 1 System SSL Cryptographic Module

As shown in Figure 2, System SSL Cryptographic Module in a z/OS Sysplex Environment, a System SSL cryptographic module may be deployed in a high availability environment where the application may in effect be instantiated on multiple z/OS system instances configured in a “clustered” environment known as a parallel sysplex. A parallel sysplex makes 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 enable session ID caching, the System SSL DLLs invoke the GSKSRVR started task to determine if an applicable SSL session has been cached. This GSKSRVR task utilizes a secure communication channel provided by the z/OS operating environment to communicate with other instances of the GSKSRVR executing on other systems in the sysplex environment.

This figure shows the client negotiating a secure connection with Application Server 1 using a full handshake. A second negotiation is performed using the cached session ID information with an identical server, Application Server 2, using a resumed handshake. The resumed handshake utilizes the session ID cache information built during the full handshake.



1 2 Application Server 1 invokes System SSL specifying sysplex session id caching to be enabled. System SSL services are invoked to negotiate a SSL session with application server 1’s client and session id is stored in the cache managed by GSKSRVR.

3 Application Server 2 invokes System SSL specifying sysplex session id caching to be enabled. System SSL services is invoked to negotiate a SSL session with the session id information from the connection in step 1 and 2.

4 5 z/OS System SSL invokes GSKSRVR to determine if an applicable session has been previously cached.
This may include an interrogation of other GSKSRVR managed caches on other z/OS sysplex members through a secure channel. If an applicable session cache entry is not found a full handshake proceeds and session cache entry is constructed. If an applicable session entry is found either locally, or within the GSKSRVR caches of other sysplex members, a resumed session handshake is performed.

A B When performing a full handshake, RSA functions available through the CEX3 cards are utilized by calling ICSF as a“passthru”.

Figure 2 System SSL Cryptographic Module in a z/OS Sysplex Environment

7 Key Management

Key Storage The System SSL library retains key material within its address space. In a typical SSL/TLS setting, private keys would be imported from a different key store. Public keys (certificates) would be distributed through other channels, such as out-of-band PKI messages.

The module provides key import and export routines to applications such that key material can be used in conjunction with cryptographic services. It is the responsibility of applications using library 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 library exists within the physical boundary of the host computer.

Key material resides in memory as clear data or in a standard key store format. The most frequently used standard formats, using passphrase-derived keys such as PKCS#12, are classified as clear-key storage according to **FIPS Pub 140-2** guidelines.

Key Generation Key Generation uses an approved RNG algorithm (specified both in **FIPS Pub 186-2** and ANSI X9.31) which is based on SHA-1. The RNG has a maximum number of internal states of 2^{160}, this maximum number reflecting the limitation of the compression function in SHA-1. RSA DSA and DH key generation algorithms use the RNG engine seeded with 20 bytes of true random data. This true random number generator extracts entropy from time measurement jitter (minute variations of clock edges). The internal TRNG engine feeds entropy on demand into the RNG; the TRNG itself maintains a running pool of samples, and provides seed if the pool passes basic entropy content checks.

DSA key generation is done according to **FIPS Pub 186-2**. RSA key generation only implements the ANSI X9.31 key generation method [3].

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 Encrypt/decrypt and Diffie-Hellman (DH) key agreement.

When using Diffie-Hellman in FIPS 140-2 mode, the allowed modulus length is 2048 bits, which provides 112 bits of encryption strength.

When using RSA Encrypt/decrypt 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.

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

The module does not output or input keys outside of the physical boundary, with the exception of secret keys that are used for key establishment. The secret keys are wrapped with RSA.

Key Protection To enforce compliance with **FIPS Pub 140-2** key management requirements on the System SSL library itself, 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. It is assumed that a unique process is allocated for each request, and that the operating system and the underlying hardware control access to the address space which contains the process that uses the module. Each instance of the cryptographic module is self-contained within a process; the library relies on such process separation and address separation to maintain confidentiality of secrets. All platforms used during **FIPS Pub 140-2** validation provide per-process protection for user data. Keys stored internally within the address range of System SSL are similarly separated logically (even if they reside in the same address space).

All keys are associated with the User role. It is the responsibility of application program developers to protect keys exported from the System SSL module.

Key Destruction Applications must destroy persistent key objects and similar sensitive information using **FIPS Pub 140-2** compliant procedures. The System SSL library itself does not destroy externally stored keys and secrets, as it does not own or discard persistent objects. Objects, when released on behalf of a caller, are erased before they are released.

8 Physical Security

The System SSL installation inherits the physical characteristics of the host running it. The System SSL library has no physical security characteristics of its own. Figure 3 illustrates an IBM zEnterprise 196 (z196) mainframe computer.

The CEX3A and CEX3C are hardware devices (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.

The CP Assist for Cryptographic Function (CPACF) accelerates the encrypting and decrypting of SSL transactions and VPN-encrypted data transfers and data-storing applications. The assist function uses a special instruction set for symmetrical clear key cryptographic encryption and encryption operations. Five special instructions are used with the cryptographic assist function.

Figure 3 IBM zEnterprise 196 (z196) Mainframe Computer

Figure 4 Crypto Express3 Card.

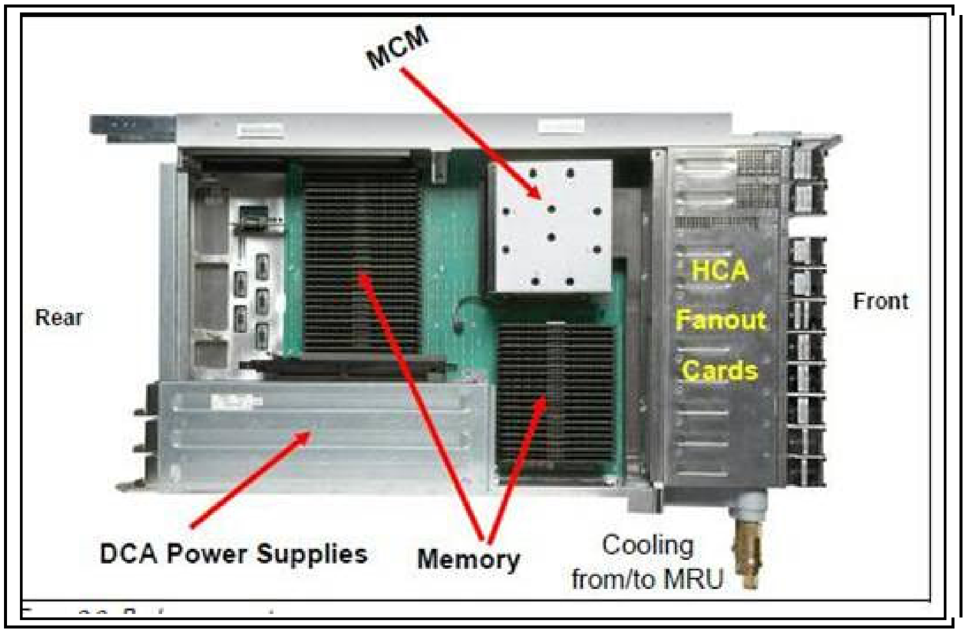
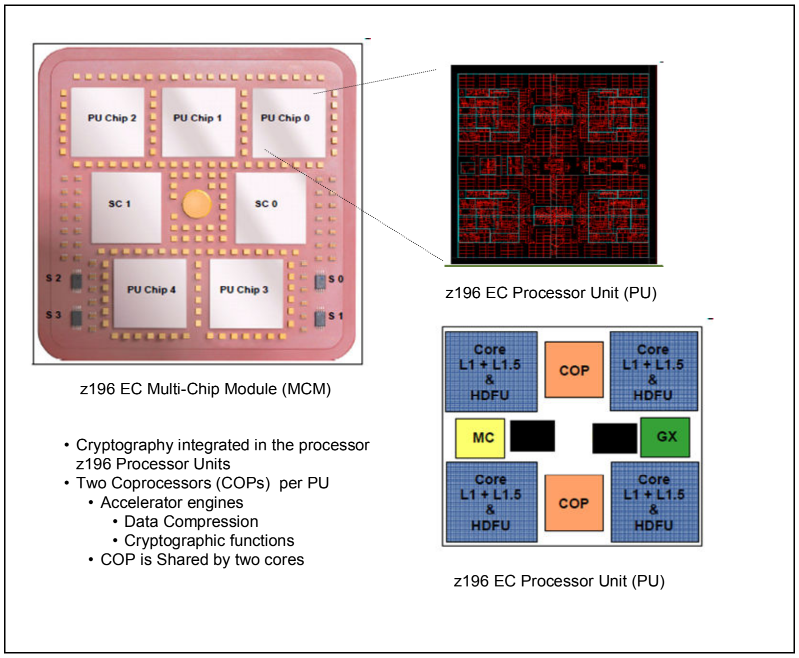


Figure 5 CPBOOK



- 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

9 EMI/EMC

EMI/EMC properties of System SSL are not meaningful for the library itself. Systems utilizing the System SSL library 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 and CEX3C cards are met by the card's FCC Class B rating.

10 Self-Tests

10.1 System SSL Module

The System SSL library 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. If any of these tests fail, the module will terminate the loading process. The module cannot be used in this state.

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 all other System SSL modules being utilized.

Algorithm known answer tests are performed when the application invokes gsk _fips_state_set to define FIPS 140-2 compliant mode and prior to any cryptographic algorithms being executed in FIPS 140-2 mode.

The module tests the following cryptographic algorithms: AES, Triple DES, RSA (sign/verify, encrypt/decrypt), DSA (sign/verify), SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, HMAC-SHA-1, and the RNG.

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

1. Integrity test on library, using RSA/SHA-256
2. Known-answer tests on algorithms, from integrity-verified binary.

The integrity check process covers all constituent DLLs and executables. DLLs and executables are individually signed and verified.

Startup Recovery If any of the startup self-tests fail, System SSL will terminate FIPS 140-2 processing.

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

If the RNG 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 library instance.

Similar to the RNG, high-entropy seed extracted by the TRNG is checked for repeated blocks, before seeding the RNG. 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 private key. The private key structure always contains either the data of the corresponding public key or information sufficient for computing the corresponding public key.

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 reinitializing the library instance.

Invoking FIPS 140-2 self-tests on demand. If a user can access System SSL services, the library has passed its integrity and power-up self tests. During regular operations, an application can invoke the gsk_perform_kat function to repeat the known answer tests on demand for the algorithms within the System SSL library DLLs. If these checks pass, the module is working properly.

If a KAT failure is encountered, 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 reinitializing the library instance.

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

10.2 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 executes 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.

11 Operational Requirements (Officer/User Guidance)

11.1 Module Configuration for FIPS 140-2 Compliance

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

- Administrators and users of System SSL must verify that the correct Security Manager Profiles have been defined to ensure that startup integrity tests are performed. Each executable and DLL 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 (through the gsk_fips state_query query).

- For applications exploiting Elliptic Curve Cryptography, ICSF must be configured to execute in FIPS 140-2 mode.

- Applications and libraries using System SSL features must observe **FIPS Pub 140-2** rules for key management and provide their own self-tests. If users of System SSL perform non-FIPS 140-2 compliant operations, they must indicate to the library that it is no longer in FIPS 140-2 mode (through the gsk_fips_state_set call).

- 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 library 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.)

- An instance of the module must not be used by multiple callers simultaneously such that they might interfere with each other. Note that for keys retained in caller-provided storage, this requirement is automatically met if the OS provides sufficient process separation (since the ownership of each memory region, therefore, each object, is uniquely determined.)

- Applications using System SSL services must verify that ownership of keys is not compromised, and keys are not shared between different users of the calling application.

  Note that this requirement is not enforced by the System SSL library itself, but by the application providing the keys to System SSL.

- Applications utilizing System SSL services must avoid using non-approved algorithms or modes of operation. If not feasible, the applications must indicate that they use utilize non-approved cryptographic services.

- To be in FIPS 140-2 mode, the System SSL 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 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.

11.2 Determining Mode of Operation

The module provides a dedicated status query (gsk_fips_state_query). This function will start indicating FIPS 140-2 mode after all self-tests are successfully completed. Callers may switch into non-FIPS 140-2 mode through a call to the status control function, gsk_fips_state_set.

To return to FIPS 140-2 mode after a mode change, the application must re-instantiate the library or executable (i.e., reload it).

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.

11.3 Testing/Physical Security Inspection Recommendations

In addition to automatic tests, which are described elsewhere in this document, System SSL users may invoke FIPS 140-2 mode self-tests at any time. These self-tests are initiated through a dedicated function (gsk_perform_kat), which is invoked automatically at startup. Continuous tests reside within their respective functions and are called implicitly during the function processing. These tests are not observable unless a failure is detected.

Apart from prudent security practice of server applications and those of security-critical embedded systems, no further restrictions are placed on hosts utilizing these services.

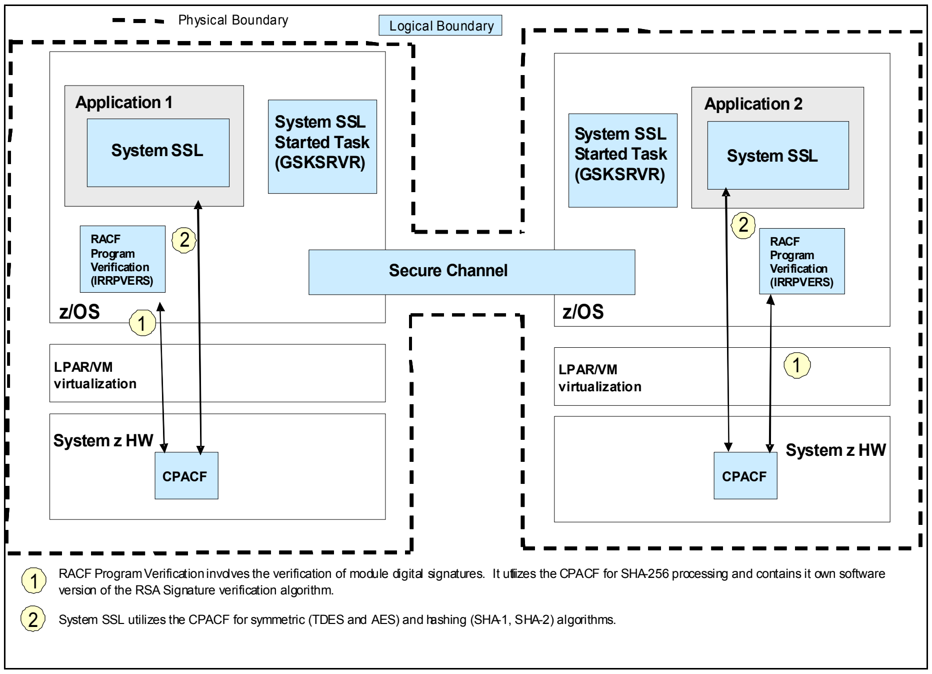
12 Mitigation of Other Attacks

The Mitigation of Other attacks security section of FIPS 140-2 is not applicable to the System SSL cryptographic module.

13 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 IBM zEnterprise 196 (z196) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863

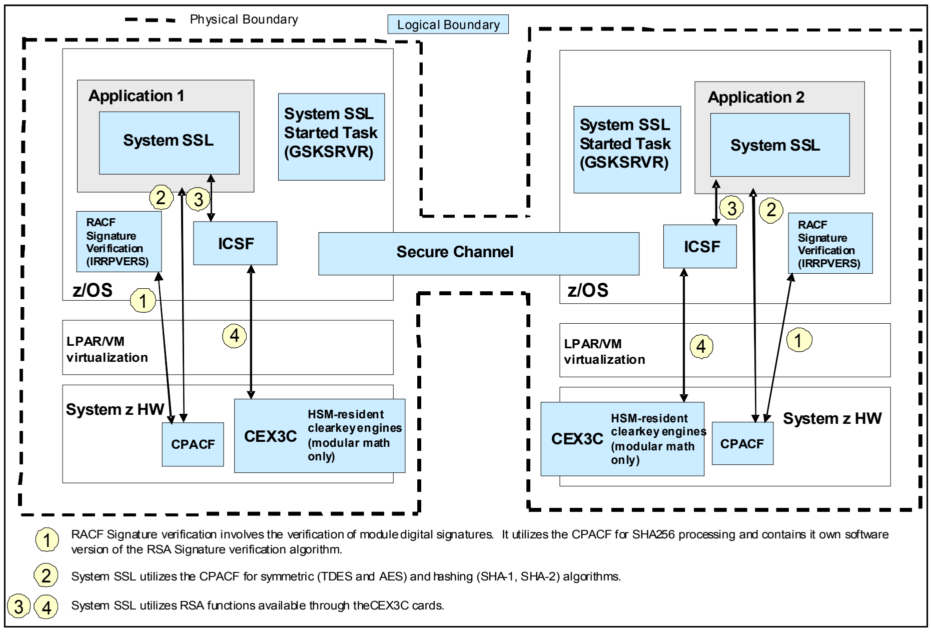


1 RACF Program Verification involves the verification of module digital signatures. It utlizes the CPACF for SHA-256 processing and contains it own software version of the RSA Signature verification algorithm.

2 System SSL utilizes the CPACF for symmetric (TDES and AES) and hashing (SHA-1, SHA-2) algorithms.

Figure 7 Validated Configuration with CPACF only

Figure 8 illustrates IBM zEnterprise 196 (z196) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and Crypto Express3 cards (Coprocessor (CEX3C))


Physical Boundary
Logical Boundary
Application 1
System SSL
System SSL Started Task (GSKSRVR)
RACF Signature Verification (IRRPVERS)
ICSF
Secure Channel
z/OS
LPAR/VM virtualization
System z HW
CPACF
CEX3C
HSM-resident clearkey engines (modular math only)
Application 2
System SSL
System SSL Started Task (GSKSRVR)
ICSF
RACF Signature Verification (IRRPVERS)
z/OS
LPAR/VM virtualization
System z HW
CEX3C
HSM-resident clearkey engines (modular math only)
CPACF

1 RACF Signature verification involves the verification of module digital signatures. It utilizes the CPACF for SHA256 processing and contains it own software version of the RSA Signature verification algorithm.

2 System SSL utilizes the CPACF for symmetric (TDES and AES) and hashing (SHA-1, SHA-2) algorithms.

3 4 System SSL utilizes RSA functions available through theCEX3C cards.

Figure 8 Validated Configuration with CPACF and CEX3C cards

Figure 9 illustrates IBM zEnterprise 196 (z196) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and Crypto Express3 cards (Accelerator (CEX3A)) configuration.

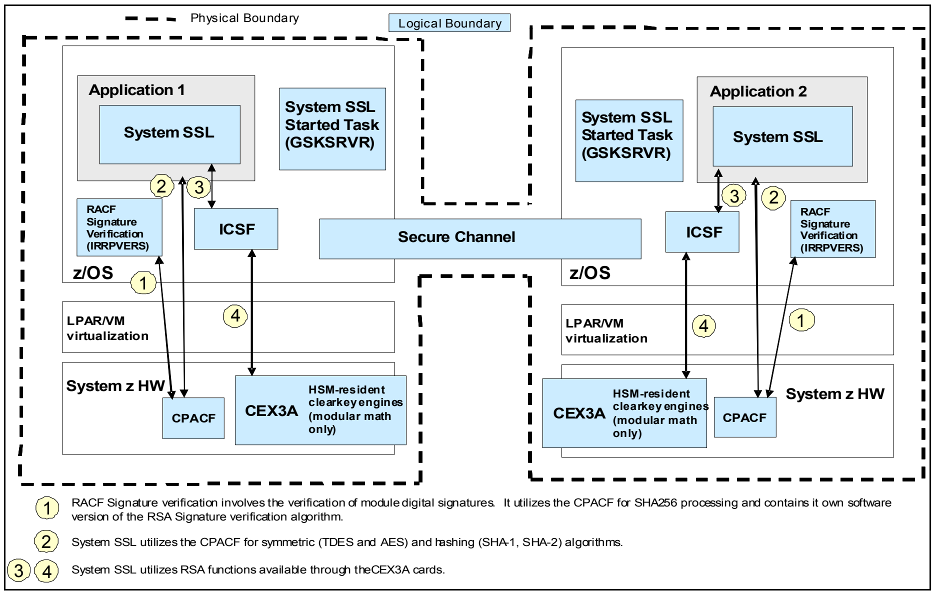

Physical Boundary
Logical Boundary
Application 1
System SSL
System SSL Started Task (GSKSRVR)
RACF Signature Verification (IRRPVERS)
ICSF
Secure Channel
z/OS
LPAR/VM virtualization
System z HW
CPACF
CEX3A
HSM-resident clearkey engines (modular math only)
Application 2
1
2
3
4
1 RACF Signature verification involves the verification of module digital signatures. It utilizes the CPACF for SHA256 processing and contains it own software version of the RSA Signature verification algorithm.
2 System SSL utilizes the CPACF for symmetric (TDES and AES) and hashing (SHA-1, SHA-2) algorithms.
3 4 System SSL utilizes RSA functions available through theCEX3A cards.

Figure 9 Validated Configuration with CPACF and CEX3A cards

Figure 10 illustrates IBM zEnterprise 196 (z196) with CP Assist for Cryptographic Functions DES/TDES Enablement Feature 3863 and Crypto Express3 cards (Coprocessor (CEX3C) and Accelerator (CEX3A)) configuration.

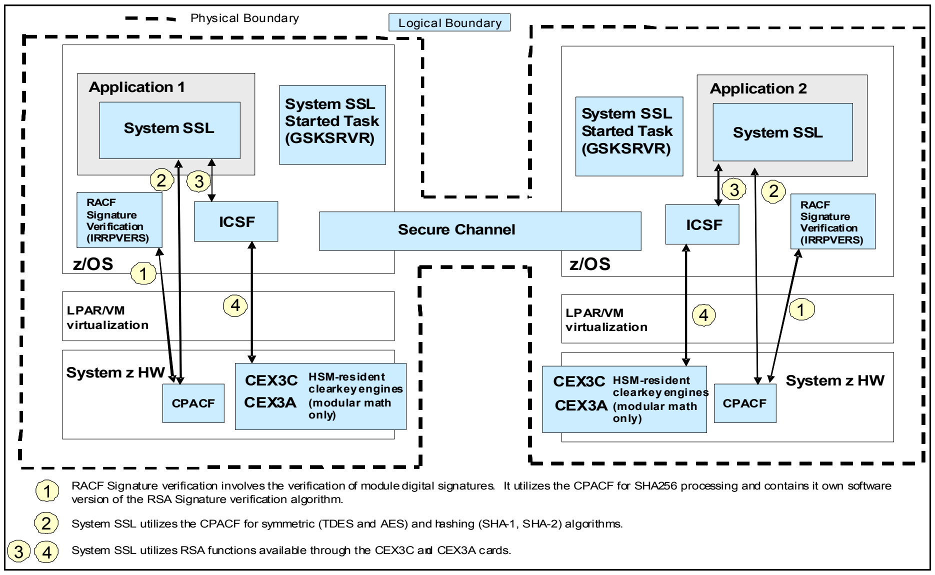

Physical Boundary
Logical Boundary
Application 1
System SSL
System SSL Started Task (GSKSRVR)
RACF Signature Verification (IRRPVERS)
ICSF
Secure Channel
z/OS
LPAR/VM virtualization
System z HW
CPACF
CEX3C
CEX3A
HSM-resident clearkey engines (modular math only)
Application 2
1
2
3
4
1 RACF Signature verification involves the verification of module digital signatures. It utilizes the CPACF for SHA256 processing and contains it own software version of the RSA Signature verification algorithm.
2 System SSL utilizes the CPACF for symmetric (TDES and AES) and hashing (SHA-1, SHA-2) algorithms.
3 4 System SSL utilizes RSA functions available through the CEX3C ad CEX3A cards.

Figure 10 Validated Configuration with CPACF, CEX3C and CEX3A cards

14 Application Programming Interfaces (APIs)

The following Services (APIs) in Table 8 can be executed by the user. 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-MD5
- RSA sign/verify, encrypt/decrypt, key generation
- DSA sign/verify, key parameter and key generation
- Diffie-Hellman key agreement and key generation
- RNG
- ECDSA sign/verify

Table 8 System SSL Module Services (APIs)

SSL Service Name	Function Description
gsk_attribute_get_buffer	Returns an attribute buffer value for an SSL environment or SSL connection
gsk_attribute_get_cert_info	Returns local or partner certificate from an SSL handshake
gsk_attribute_get_data	Returns information to the application about data in the certificate request SSL handshake message
gsk_attribute_get_enum	Returns an attribute enumerated value for an SSL environment or connection
gsk_attribute_get_numeric_value	Returns an attribute numeric value for an SSL environment or connection
gsk_attribute_set_buffer	Sets an attribute value for an SSL environment or SSL connection
gsk_attribute_set_callback	Sets the application callback routines
gsk_attribute_set_enum	Sets an attribute enum for an SSL environment or SSL connection
gsk_attribute_set_numeric_value	Sets an attribute numeric value for an SSL environment or SSL connection
gsk_attribute_set_tls_extension	Sets TLS extensions for an SSL environment or SSL connection
gsk_environment_close	Closes an SSL environment
gsk_environment_init	Establishes the SSL environment
gsk_environment_open	Gets storage and initializes SSL default environment attributes
gsk_free_cert_data	Free storage associated with returned certificate
gsk_get_cert_by_label	Gets information about a certificate
gsk_get_cipher_suites	Determines the supported SSL ciphers
gsk_get_ssl_vector	Gets addresses for all SSL functions
gsk_get_update	Checks whether SAF key ring, key database file or PKCS#11 Token has changed since certificates were read into the SSL environment
gsk_list_free	Frees storage from gsk_attribute_get_data
gsk_secure_socket_close	Closes an SSL connection
gsk_secure_socket_init	SSL handshake is performed
gsk_secure_socket_misc	SSL rehandshake is performed
gsk_secure_socket_open	Gets storage and initializes SSL default connection attributes
gsk_secure_socket_read	Performs a secure SSL read
gsk_secure_socket_shutdown	Sends close notify alert message
gsk_secure_socket_write	Performs secure SSL write
gsk_strerror	Returns text string for an SSL or Certificate Management error code
Certificate Management Service Name	**Function Description**
gsk_add_record	Adds inputted record to a key or request database
gsk_change_database_password	Changes the password associated with the key database file to the inputted password
gsk_change_database_record_length	Changes the record length of the key database record
gsk_close_database	Closes the key or request database file
gsk_close_directory	Unbinds from the LDAP directory
gsk_construct_certificate	Constructs a signed X.509 certificate
gsk_construct_private_key	Constructs a private key from its component values

gsk_construct_private_key_rsa	Constructs an RSA private key from its component values
gsk_construct_public_key	Constructs a public key from its component values
gsk_construct_public_key_rsa	Constructs an RSA public key from its component values
gsk_construct_renewal_request	Constructs a certification renewal request as described in PKCS #10
gsk_construct_self_signed_certificate	Constructs a self-signed certificate
gsk_construct_signed_certificate	Constructs a signed certificate from a certificate request
gsk_copy_attributes_signers	Allocates storage and makes a copy of the inputted attribute signer info
gsk_copy_buffer	Allocates storage and makes a copy of the inputted buffer info
gsk_copy_certificate	Allocates storage and makes a copy of the inputted certificate
gsk_copy_certificate_extension	Allocates storage and makes a copy of the inputted certificate extension
gsk_copy_certification_request	Allocates storage and makes a copy of the inputted certificate request info
gsk_copy_content_info	Allocates storage and makes a copy of the inputted content info
gsk_copy_crl	Allocates storage and makes a copy of the inputted CRL
gsk_copy_name	Allocates storage and makes a copy of the inputted name
gsk_copy_private_key_info	Allocates storage and makes a copy of the inputted private key info
gsk_copy_public_key_info	Allocates storage and makes a copy of the inputted public key info
gsk_copy_record	Allocates storage and makes a copy of the inputted record
gsk_create_certification_request	Creates a certificate request (PKCS#10) using the inputted information and stores it in the request database. RSA/DSA Public/private keys are generated
gsk_create_database	Create a key or request database
gsk_create_database_renewal_request	Creates a PKCS#10 certification renewal request and adds it to the request database
gsk_create_database_signed_certificate	Creates a signed certificate as part of a set of certificates and adds it to the key database
gsk_create_renewal_request	Creates a renewal certificate request (PKCS#10) using the inputted certificate info (includes public/private keys) and stores in the request database
gsk_create_self_signed_certificate	Creates a self-signed certificate and stores it in the key database file
gsk_create_signed_certificate_record	Creates a signed certificate using the inputted CA certificate and certificate request and returns it in Base64 format
gsk_create_signed_certificate_set	Creates a signed certificate using the inputted CA certificate and info about the certificate being created and stores it in the key database file
gsk_create_signed_crl	Creates a new X.509 CRL (RFC 2459) signed by the inputted certificate
gsk_create_signed_crl_record	Creates a new X.509 CRL (RFC2459) signed by the inputted certificate
gsk_decode_base64	Decodes a Base64-encoded stream
gsk_decode_certificate	Decodes an X.509 certificate
gsk_decode_certificate_extension	Decodes an X.509 certificate extension
gsk_decode_certification_request	Decodes a PKCS#10 certificate request
gsk_decode_crl	Decodes an X.509 CRL
gsk_decode_import_certificate	Decodes certificate from DER-Encoded or PKCS#7-encoded data stream
gsk_decode_import_key	Decodes certificate and key from PKCS #12-encoded data stream
gsk_decode_name	Decodes an ASN.1 DER-encoded X.509 name
gsk_decode_private_key	Decodes an ASN.1 DER-encoded PKCS#8 private key
gsk_decode_public_key	Decodes an ASN.1 DER-encoded public key
gsk_delete_record	Deletes record from a key or request database file
gsk_dn_to_name	Converts a DN name to a X.509 name
gsk_encode_base64	Encodes binary data using Base64 encoding
gsk_encode_certificate_extension	Encodes a X.509 certificate extension
gsk_encode_export_certificate	Encodes an X.509 certificate into a DER or PKCS#7 data stream
gsk_encode_export_key	Encodes an X.509 certificate and its private key into a PKCS #12 data stream
gsk_encode_export_request	Encodes a certification renewal request as described in PKCS #10
gsk_encode_private_key	Encodes a private key (PKCS#8 format)
gsk_encode_public_key	Encodes a public key
gsk_encode_signature	Encodes an ASN.1 stream and the accompanying signature
gsk_export_certificate	Returns the specified certificate (and public key) in either DER (binary/Base64) or PKCS#7 (binary/Base64) format
gsk_export_certification_request	Returns the specified certificate request in PKCS#10 format
gsk_export_key	Returns the specified certificate and private key in PKCS#12 format
gsk_factor_private_key	Factorizes a private key into its component values
gsk_factor_private_key_rsa	Factorizes an RSA private key into its component values
gsk_factor_public_key	Factorizes a public key into its component values
gsk_factor_public_key_rsa	Factorizes an RSA public key into its component values
gsk_fips_state_query	Returns the current FIPS mode
gsk_fips_state_set	Sets either FIPS or Non-FIPS mode of operation

gsk_free_attributes_signers	Frees storage obtained for gsk_attributes_signers structure
gsk_free_buffer	Frees storage obtained for a buffer
gsk_free_certificate	Frees storage obtained for a X.509 certificate
gsk_free_certificates	Frees storage obtained for an array of X.509 certificates
gsk_free_certificate_extension	Frees storage obtained for a X.509 certificate extension
gsk_free_certification_request	Frees storage obtained for PKCS#10 certificate request
gsk_free_content_info	Frees storage obtained for PKCS#7 content information
gsk_free_crl	Frees storage obtained for a X509 certificate revocation list (CRL)
gsk_free_crls	Frees storage obtained for an array of X.509 CRLs
gsk_free_decoded_extension	Frees storage for a decoded certificate extension
gsk_free_name	Frees storage for a X.509 name
gsk_free_private_key	Frees storage allocated for private key information
gsk_free_private_key_info	Frees storage for a private key information structure
gsk_free_public_key	Frees storage allocated for public key information
gsk_free_public_key_info	Frees storage for a public key information structure
gsk_free_record	Frees storage for a database record
gsk_free_records	Frees storage for an array of database records
gsk_free_string	Frees storage for a string
gsk_free_strings	Frees storage for an array of strings
gsk_generate_key_agreement_pair	Generates a DH public/private key pair
gsk_generate_key_pair	Generates a public/private key pair
gsk_generate_key_parameters	Generates key parameters
gsk_generate_random_bytes	Generates a random byte stream
gsk_generate_secret	Generates the Diffie-Hellman shared secret
gsk_get_certificate_info	Returns requested certificate information for an x.509 certificate
gsk_get_certificate_algorithms	Gets the public key and certificate signature algorithms from a certificate
gsk_get_cms_vector	Obtains the address of the CMS function vector
gsk_get_default_key	Obtains the default certificate record in a key database file, SAF key ring or PKCS#11 Token
gsk_get_default_label	Gets the default certificate record in a key database file, SAF key ring or PKCS#11 Token
gsk_get_directory_certificates	Gets the certificates stored in a LDAP directory for the inputted subject name
gsk_get_directory_crls	Gets the CRLs stored in a LDAP directory for the inputted issuer
gsk_get_directory_enum	Gets an enumerated value from an LDAP directory
gsk_get_ec_parameters_info	Get the named curve type and key size for EC domain parameters
gsk_get_record_by_id	Gets a record from a key database or request database using the record identifier
gsk_get_record_by_index	Gets a record from a key database or request database using the record index
gsk_get_record_by_label	Gets a record from a key database or request database using the record label
gsk_get_record_by_subject	Gets the records from a key database or request database using the certificate subject name
gsk_get_record_labels	Gets the labels for all records in a key or request database
gsk_get_update_code	Returns the current update code
gsk_import_certificate	Imports a certificate into the key database file
gsk_import_key	Imports a certificate chain and private key into the key database file
gsk_make_content_msg	Creates a PKCS#7 content information message
gsk_make_data_content	Creates PKCS#7 data content information from application data
gsk_make_data_msg	Creates a PKCS#7 data message from application data
gsk_make_encrypted_data_content	Creates PKCS#7 encrypted data content information. Not supported in FIPS mode
gsk_make_encrypted_data_msg	Creates a PKCS$7 encrypted data message from application data. Not supported in FIPS mode
gsk_make_enveloped_data_content	Creates PKCS#7 Enveloped Data content information
gsk_make_enveloped_data_content_extended	Creates a PKCS#7 EnvelopedData content information
gsk_make_enveloped_data_msg	Creates a PKCS#7 EnvelopedData message from application data
gsk_make_enveloped_data_msg_extended	Creates a PKCS#7 EnvelopedData message from application data
gsk_make_signed_data_content	Creates a PKCS#7 SignedData content information
gsk_make_signed_data_content_extended	Creates a PKCS#7 SignedData content information
gsk_make_signed_data_msg	Creates a PKCS#7 SignedData message from application data
gsk_make_signed_data_msg_extended	Creates a PKCS#7 SignedData message from application data
gsk_make_wrapped_content	Wraps the supplied content information in an ASN.1 sequence
gsk_mktime	Converts year/month/day time value to number of seconds since the POSIX epoch

gsk_name_compare	Compares two X.509 names
gsk_name_to_dn	Converts an X.509 name to a distinguished name string
gsk_open_database	Opens a key or request database
gsk_open_database_using_stash_file	Opens a key or request database using a stash file the database password
gsk_open_directory	Opens an LDAP directory
gsk_open_keyring	Opens a SAF key ring or PKCS#11 Token and constructs a read-only version of the certificates in storage
gsk_perform_kat	Performs known answer tests against FIPS approved/allowed algorithms
gsk_query_crypto_level	Returns the available cryptographic levels
gsk_query_database_label	Determines if a label exists in the in storage copy of a key database file, SAF key ring or PKCS#11 Token
gsk_query_database_record_length	Returns the record length for a key database file
gsk_rdtime	Converts the number of seconds since the EPOCH to year/month/day
gsk_read_content_msg	Processes a PKCS#7 message and returns the content information
gsk_read_data_content	Processes a PKCS#7 Data content information and returns the application data
gsk_read_data_msg	Processes a PKCS#7 message and returns the application data
gsk_read_encrypted_data_content	Processes PKCS#7 EncryptedData content information. Not supported in FIPS mode
gsk_read_encrypted_data_msg	Processes PKCS#7 EncryptedData message and returns the decrypted message content. Not Supported in FIPS mode
gsk_read_enveloped_data_content	Processes the EnvelopedData content information
gsk_read_enveloped_data_content_extended	Processes the EnvelopedData content information
gsk_read_enveloped_data_msg	Processes a PKCS#7 EnvelopedData message
gsk_read_enveloped_data_msg_extended	Processes a PKCS#7 EnvelopedData message
gsk_read_signed_data_content	Processes PKCS#7 SignedData content information
gsk_read_signed_data_content_extended	Processes PKCS#7 SignedData content information
gsk_read_signed_data_msg	Processes PKCS#7 SignedData message
gsk_read_signed_data_msg_extended	Processes PKCS#7 SignedData message
gsk_read_wrapped_content	Processes the ASN.1 sequence containing the encoded content information
gsk_receive_certificate	Receives X.509 certificates package
gsk_replace_record	Replaces a record in the key database file
gsk_set_default_key	Sets the default key (certificate) in a key database file
gsk_set_directory_enum	Sets an enumerated value for an LDAP directory
gsk_sign_certificate	Signs a certificate using the supplied RSA or DSA private key
gsk_sign_crl	Signs a X.509 CRL using the supplied RSA or DSA private key
gsk_sign_data	Signs a data stream using the supplied RSA or DSA private key
gsk_validate_certificate	Validates the inputted certificate chain
gsk_validate_certificate_mode	Validate the inputted certificate chaing according to RFC2459 or RFC3280
gsk_validate_hostname	Validates a host certiifcate against the supplied hostname
gsk_validate_server	Validates the inputted hostname against the inputted certificate
gsk_verify_certificate_signature	Verifies the signature for an X.509 certificate. (RSA or DSA)
gsk_verify_crl_signature	Verifies the signature for an X.509 CRL (RSA or DSA)
gsk_verify_data_signature	Verifies the signature (RSA or DSA)
Deprecated SSL Service Name	**Function Description**
gsk_free_memory	Frees storage obtained by gsk_get_dn_ by_label
gsk_get_cipher_info	Determines the supported SSL ciphers
gsk_get_dn_by_label	Returns the distinguished name in the certificate identified by label
gsk_initialize	Established the SSL
gsk_secure_soc_close	Returns storage associated with an SSL connection
gsk_secure_soc_init	Performs SSL handshake
gsk_secure_soc_read	Performs a secure SSL read
gsk_secure_soc_reset	SSL rehandshake is performed
gsk_secure_soc_write	Performs a secure SSL write
gsk_srb_initialize	Initializes an SRB
GSKSRBRD	SSL Secure read in SRB
GSKSRBWT	SSL secure write in SRB
gsk_uninitialize	Returns storage for an SSL environment
gsk_user_set	Sets the application callback routines for session id caching
Other Service Name	**Function Description**
GSKSRVR	GSKSRVR is started task that provides sysplex session cache support, dynamic trace support and notification through messages when an application has switched from using hardware to software for its cryptographic support.

gskkyman	gskkyman is System SSL's certificate utility. It provides certificate management capabilities for both key database files and PKCS#11 tokens
R_PgmSignVer	Provides support for verifying digital certificates associated with a load module

15 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.

16 References

[1] z/OS Cryptographic Services Secure Sockets Layer Programming (SC24-5901-09)

[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)

17 Trademarks