z/OS
Version 2.Release 4

*Cryptographic Services*
*Integrated Cryptographic Service Facility*
*Overview*

IBM

**Note**

Before using this information and the product it supports, read the information in "Notices" on page 83.

This edition applies to ICSF FMID HCR77D1 and Version 2 Release 4 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2021-06-22

# Contents

# Figures

# Tables

x

# About this information

This information contains overview and planning information for the z/OS Integrated Cryptographic Service Facility (ICSF). The z/OS Cryptographic Services includes these components:

- z/OS Integrated Cryptographic Service Facility (ICSF)
- z/OS System Secure Socket Level Programming (SSL)
- z/OS Public Key Infrastructure Services (PKI)

ICSF is a software element of z/OS that works with hardware cryptographic features and the Security Server (RACF) to provide secure, high-speed cryptographic services in the z/OS environment. ICSF provides the application programming interfaces by which applications request the cryptographic services. The cryptographic feature is secure, high-speed hardware that performs the actual cryptographic functions.

The cryptographic hardware features available to your applications depend on the server.

## ICSF features

ICSF provides support for:

- The ANSI Data Encryption Algorithm (DES) and Advanced Encryption Standard (AES) encryption and decryption
- DES key management and transport
- AES key management and transport
- Financial services including PINs, payment card industry transactions and ATMs
- Public key operations including key generation, digital signatures and wrapping symmetric keys for transport
- MAC and hash generation
- Acceleration of handshake and frame encryption for SSL
- PKCS #11 API

## Who should use this information

This information is for chief information officers, information system executives, and information security professionals and auditors. Installation managers and security administrators who are responsible for planning the data security strategy for their installation will also find this information to be helpful. This publication applies to installations that have z/OS with ICSF and a hardware cryptographic feature installed.

## How to use this information

The major topics are:

- Chapter 1, "Introducing cryptography and ICSF," on page 1 introduces the general subject of cryptography, and describes why ICSF may be right for your installation.
- Chapter 2, "Solving your business needs with ICSF," on page 9 describes how ICSF can help your business.
- Chapter 3, "Application Programming Interfaces and key management," on page 23 describes the cryptographic callable services available with ICSF and the basic concepts of managing cryptographic keys.
- Chapter 4, "Using ICSF with other cryptographic products," on page 39 describes how ICSF relates to other cryptographic products.

- Chapter 5, "Planning for the Integrated Cryptographic Service Facility," on page 43 identifies the system facilities and system resources that ICSF requires and presents guidelines and suggestions to help you when you plan for installing, operating, and migrating ICSF.
- Appendix A, "Standards," on page 55 provides a list of International and USA standards for the Crypto Express adapters and ICSF.
- Appendix B, "Summary of callable service support by hardware configuration," on page 63 summarizes ICSF callable services by configuration.

# Where to find more information

This topic describes what contains ICSF information.

## The ICSF library

The ICSF library includes these publications:

- *z/OS Cryptographic Services ICSF Overview*

  This publication provides an introduction to ICSF, an overview of cryptography, and planning information.

- *z/OS Cryptographic Services ICSF Administrator's Guide*

  See this for information on managing cryptographic keys. It describes the tasks of entering CCA and PKCS #11 master keys, changing master keys, managing cryptographic key data sets, using the key generator utility program and viewing the status of ICSF and cryptographic coprocessors.

- *z/OS Cryptographic Services ICSF System Programmer's Guide*

  See this for information on initialization, customization, migration, and problem diagnosis.

- *z/OS Cryptographic Services ICSF Application Programmer's Guide*

  See this for information on writing application programs that use the callable services that are provided by ICSF to access cryptographic functions. These callable services can be used in high-level languages such as C, COBOL, FORTRAN, and PL/I, as well as in Assembler.

- *z/OS Cryptographic Services ICSF Messages*

  See this for explanations of messages that are produced by ICSF, and for the routing and descriptor codes for those messages.

- *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*

  See this for information about the PKCS #11 support provided by ICSF.

- *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*

  This information is available with the optional Trusted Key Entry (TKE) workstation and explains how to install and run the TKE workstation for key distribution.

These publications contain additional ICSF information:

- *z/OS MVS System Codes*

  This describes reason codes for ICSF X'18F' abend code.

- *z/OS MVS Initialization and Tuning Guide*
- *z/OS MVS Initialization and Tuning Reference*
- *z/OS MVS Programming: Callable Services for High-Level Languages*
- *z/OS MVS Programming: Authorized Assembler Services Guide*
- *z/OS MVS Programming: Extended Addressability Guide*
- *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*
- *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*
- *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU*

- *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*

# Related publications

- *IBM Encryption Facility for z/OS: Using Encryption Facility for OpenPGP*
- *z/OS Security Server RACF Auditor's Guide*
- *z/OS Security Server RACF Command Language Reference*
- *z/OS Security Server RACF Security Administrator's Guide*
- *z/OS Security Server RACF Macros and Interfaces*
- *z/OS Security Server RACF System Programmer's Guide*
- *IBM Distributed Key Management System, Installation and Customization Guide*
- *IBM Common Cryptographic Architecture: Cryptographic Application Programming Interface Reference*
- *IBM ES/3090 Processor Complex PR/SM Planning Guide*
- *IBM Security Architecture: Securing the Open Client/Server Distributed Enterprise*
- *VTAM Programming for LU 6.2*
- *RSA's Frequently Asked Questions About Today's Cryptography*, available at RSA (www.rsa.com).
- *Applied Cryptography*, Second Edition, by Bruce Schneier

# Information on other IBM cryptographic products

- *IBM 4767 Specification Sheet*
- *IBM 4767 PCIe Cryptographic Coprocessor Safety Notices*
- *IBM 4767 Warranty Information Flyer*
- *IBM 4765 Specification Sheet*
- *IBM 4765 PCIe Cryptographic Coprocessor Safety Notices*
- *IBM 4765 Warranty Information Flyer*
- *CCA Basic Services Reference and Guide for the IBM 4767 and IBM 4765 PCIe Cryptographic Coprocessors*

# IBM Crypto Education

The IBM Crypto Education (community.ibm.com/community/user/ibmz-and-linuxone/groups/community-home?CommunityKey=6593e27b-caf6-4f6c-a8a8-10b62a02509c) community provides detailed explanations and samples pertaining to IBM cryptographic technology.

# How to send your comments to IBM

We invite you to submit comments about the z/OS® product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

**Important:** If your comment regards a technical question or problem, see instead "If you have a technical problem" on page xv.

Submit your feedback by using the appropriate method for your type of comment or question:

**Feedback on z/OS function**
If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

**Feedback on IBM® Knowledge Center function**
If your comment or question is about the IBM Knowledge Center functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Knowledge Center Support at ibmkc@us.ibm.com.

**Feedback on the z/OS product documentation and content**
If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: z/OS ICSF Overview, SC14-7505-09
- The section title of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive authority to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

# If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

# Summary of changes

ICSF is an element of z/OS, but provides independent ICSF releases as web deliverables. These web deliverables are identified by their FMID. Each release of z/OS includes a particular ICSF FMID level as part of its base.

ICSF publications can be obtained from:

- The Resource Link home page (www.ibm.com/servers/resourcelink). (Select Publications and then select the release that you are interested in under ICSF Publications by FMID.)
- IBM z/OS downloads (www.ibm.com/systems/z/os/zos/downloads) for Cryptographic Support.

This document contains terminology, maintenance, and editorial changes to improve consistency and retrievability. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

## Changes made in Cryptographic Support for z/OS V2R2 - z/OS V2R4 (FMID HCR77D1)

This document contains information previously presented in *z/OS ICSF Overview*, SC14-7505-08.

This document is for ICSF FMID HCR77D1. This release of ICSF runs on z/OS V2R2, z/OS V2R3, and z/OS V2R4 and only on zSeries hardware.

The most recent updates are listed at the top of each section.

### New

**June 2020 refresh**

- "CRYSTALS-Dilithium private and public keys" on page 29 (APAR OA58880)
- "Quantum-safe cryptography" on page 38 (APAR OA58880)

**Prior to the June 2020 refresh**

- Support for propagating KDS updates in a GDPS environment (APAR OA56203).
- Information about IBM z15.

### Changed

**May 2021 refresh**

- "Using Message Authentication Codes" on page 15 (APAR OA60318)
- "Control vectors" on page 25 (APAR OA60318)
- "DES key token wrapping" on page 27 (APAR OA60318)

**December 2020 refresh**

- Appendix B, "Summary of callable service support by hardware configuration," on page 63 (APAR OA60317)

**October 2020 refresh**

- Appendix B, "Summary of callable service support by hardware configuration," on page 63 (APAR OA59593)

**June 2020 refresh**

- "Server hardware" on page 45 (APAR OA58880)
- Appendix B, "Summary of callable service support by hardware configuration," on page 63 (APAR OA58880)

**Prior to the June 2020 refresh**

- "Security" on page 49

### Deleted

PCI Cryptographic Coprocessor information.

# Changes made in Cryptographic Support for z/OS V2R2 - z/OS V2R3 (FMID HCR77D0)

This document contains information previously presented in *z/OS ICSF Overview*, SC14-7505-07.

This document is for ICSF FMID HCR77D0. This release of ICSF runs on z/OS V2R2 and z/OS V2R3 and only on zSeries hardware.

The most recent updates are listed at the top of each section.

### New

- "Dynamic service update" on page 17

### Changed

- "ATM remote key loading" on page 8 (APAR OA57089)
- Appendix A, "Standards," on page 55 (APAR OA57089)
- Appendix B, "Summary of callable service support by hardware configuration," on page 63 (APAR OA57089).
- "System requirements" on page 43

### Deleted

No content was removed from this information.

# Changes made in Cryptographic Support for z/OS V2R1 - z/OS V2R3 (FMID HCR77C1)

This document contains information previously presented in *z/OS ICSF Overview*, SC14-7505-06.

This document is for ICSF FMID HCR77C1. This release of ICSF runs on z/OS V2R1, V2R2, and V2R3 and only on zSeries hardware.

The most recent updates are listed at the top of each section.

### New

- Information about IBM z14 and IBM z14 ZR1.

- "PCI-HSM compliance" on page 20.
- "Cryptographic usage tracking" on page 21.
- Crypto Express6 adapter (CEX6C, CEX6P, or CEX6A) was added to "Cryptographic hardware features" on page 44.

## Changed

- "Using Personal Identification Numbers (PINs) for personal authentication" on page 14 (APAR OA55184).
- "Using Message Authentication Codes" on page 15 (APAR OA55184).
- "Callable services" on page 23 (APAR OA55184).
- "Types of DES keys" on page 25 (APAR OA55184).
- "DES key token wrapping" on page 27 (APAR OA55184).
- Appendix B, "Summary of callable service support by hardware configuration," on page 63 (APAR OA55184).
- "Using RMF and SMF to monitor z/OS ICSF events" on page 17.
- "Protected-key CPACF" on page 35.
- "Cryptographic hardware features" on page 44.
- "Server hardware" on page 45.

## Deleted

No content was removed from this information.

# Changes made in Cryptographic Support for z/OS V2R1 - z/OS V2R2 (FMID HCR77C0)

This document contains information previously presented in *z/OS ICSF Overview*, SC14-7505-05.

This document is for ICSF FMID HCR77C0. This release of ICSF runs on z/OS z/OS V2R1 and z/OS V2R2 and only on zSeries hardware.

The most recent updates are listed at the top of each section.

## New

- "Auditing ICSF actions" on page 21.

## Changed

- "Protected-key CPACF" on page 35 was updated.
- "Cryptographic Key Data Set (CKDS)" on page 31 was updated.
- "Public Cryptographic Key Data Set (PKDS)" on page 33 was updated.
- Appendix B, "Summary of callable service support by hardware configuration," on page 63 was updated.

## Deleted

No content was removed from this information.

# Chapter 1. Introducing cryptography and ICSF

The Internet is rapidly becoming the basis for electronic commerce. More businesses are automating their data processing operations. Online databases are becoming increasingly large and complex. Many businesses transmit sensitive data on open communication networks and store confidential data offline. Every day the potential for unauthorized persons to access sensitive data increases.

To achieve security in a distributed computing environment, a combination of elements must work together. A security policy should be based on an appraisal of the value of data and the potential threats to that data. This provides the foundation for a secure environment.

IBM has categorized these security functions according to International Organization for Standardization (ISO) standard 7498-2:

- Identification and authentication - includes the ability to identify users to the system and provide proof that they are who they claim to be.
- Access control - determines which users can access which resources.
- Data confidentiality - protects an organization's sensitive data from being disclosed to unauthorized persons.
- Data integrity - ensures that data is in its original form and that nothing has altered it.
- Security management - administers, controls, and reviews a business security policy.
- Nonrepudiation - assures that the appropriate individual sent the message.

Only cryptographic services can provide the data confidentiality and the identity authentication that is required to protect business commerce on the Internet.

## What is cryptography?

Cryptography includes a set of techniques for scrambling or disguising data. The scrambled data is available only to someone who can restore the data to its original form. The purpose is to make data unintelligible to unauthorized persons, but readily decipherable to authorized persons. Cryptography deals with several processes:

- **Enciphering** is converting *plaintext*, which is intelligible, into *ciphertext*, which is not intelligible. Enciphering is also called encrypting.
- **Deciphering** is converting ciphertext back into plaintext. Deciphering is also called decrypting.
- **Hashing** involves using a one-way calculation to condense a long message into a compact bit string, or message digest.
- **Generating and verifying digital signatures** involves encrypting a message digest with a private key to create the electronic equivalent of a handwritten signature. Both a handwritten signature and a digital signature verify the identity of the signer and cannot be forged.

  Digital signatures also serve to ensure that nothing has altered the signed document since it was signed.

The growth of distributed systems and the increasing use of the Internet have resulted in the need for increased data security. Cryptography provides a strong, economical basis for keeping data confidential and for verifying data integrity. Cryptography is already playing a critical and expanding role in electronic commerce and electronic mail services. Emerging markets that require secure data transmission and the authentication of the sender are already relying on cryptography.

### The basic elements of a cryptographic system

Most practical cryptographic systems combine two elements:

- A process or algorithm which is a set of rules that specify the mathematical steps needed to encipher or decipher data.

**1**

- A cryptographic key (a string of numbers or characters), or keys. The algorithm uses the key to select one relationship between plaintext and ciphertext out of the many possible relationships the algorithm provides. The selected relationship determines the composition of the algorithm's result.

ICSF supports two main types of cryptographic processes:

- Symmetric, or secret key, algorithms, in which the same key value is used in both the encryption and decryption calculations.
- Asymmetric, or public key, algorithms, in which a different key is used in the decryption calculation than was used in the encryption calculation.

## Secret key cryptography

Secret key cryptography uses a conventional algorithm such as the Data Encryption Standard (DES) algorithm or the Advanced Encryption Standard (AES) algorithm that are supported by ICSF. Another term for secret key cryptography is symmetric cryptography. To have intelligent cryptographic communications between two parties who are using a conventional algorithm, this criteria must be satisfied:

- Both parties must use the same cryptographic algorithm.
- The cryptographic key that the sending party uses to encipher the data must be available to the receiving party to decipher the data.

Figure 1 on page 2 is a simplified illustration of the cryptographic components that are needed to encipher and decipher data in a secret key cryptographic system. In this system, Tom and Linda have established a secure communications channel by sharing a secret key. Tom enciphers the plaintext by using the algorithm and the secret key before sending it to Linda. When she receives the ciphertext, Linda deciphers it using the same algorithm and the same secret key. In a secret key system, it is critically important to maintain the secrecy of the shared key.
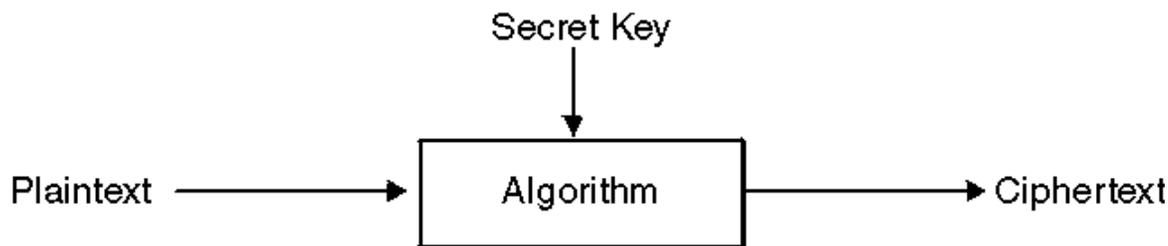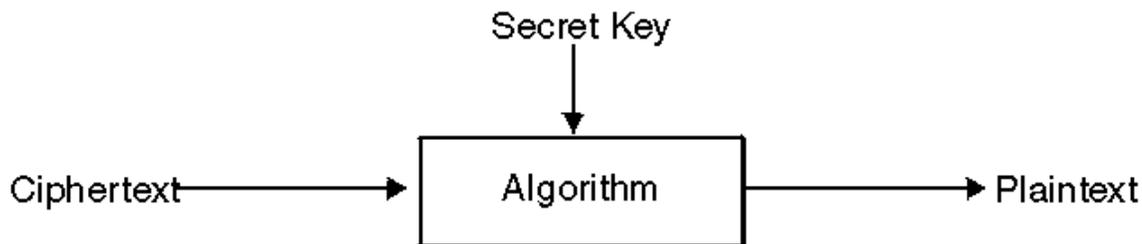


Figure 1. Enciphering and deciphering data in a secret key system

## Public key cryptography

Each party in a public key cryptography system has a pair of keys. One key is public and is published, and the other key is private. Another term for public key cryptography is asymmetric cryptography because the public key and private key are not identical. The sending party looks up the receiving party's public key and uses it to encipher the data. The receiving party then uses its private key to decipher the data. In a public key system, it is critically important to maintain the secrecy of the private key.

Public key cryptography requires complex mathematical calculations. For this reason, these types of systems are not used for enciphering messages or large amounts of data. They are, however, used to encipher and decipher symmetric keys that are transported between two systems.

Public key cryptography systems are often used to generate and verify digital signatures on electronic documents. The sender uses his or her private key to generate the digital signature. The receiver then uses the sender's public key to verify the identity of the sender. On the emerging information highway, the digital signature replaces the handwritten signature as a legal proof of authenticity. Digital signatures are the principal mechanism in any system of nonrepudiation.

Figure 2 on page 3 shows an example of a nonrepudiation system that uses digital signatures. Linda sends her broker Tom an electronic order to buy 100 shares of IBM stock. The electronic transmission application on Linda's system attaches Linda's digital signature to the order before sending the order to Tom. Linda's digital signature provides Tom with proof that Linda sent the order. When Tom receives the purchase order, an acknowledgment of his receipt, including his own digital signature, is returned to Linda. This receipt serves as proof that Tom received the order. Nonrepudiation is critical for the security of electronic data interchange (EDI).
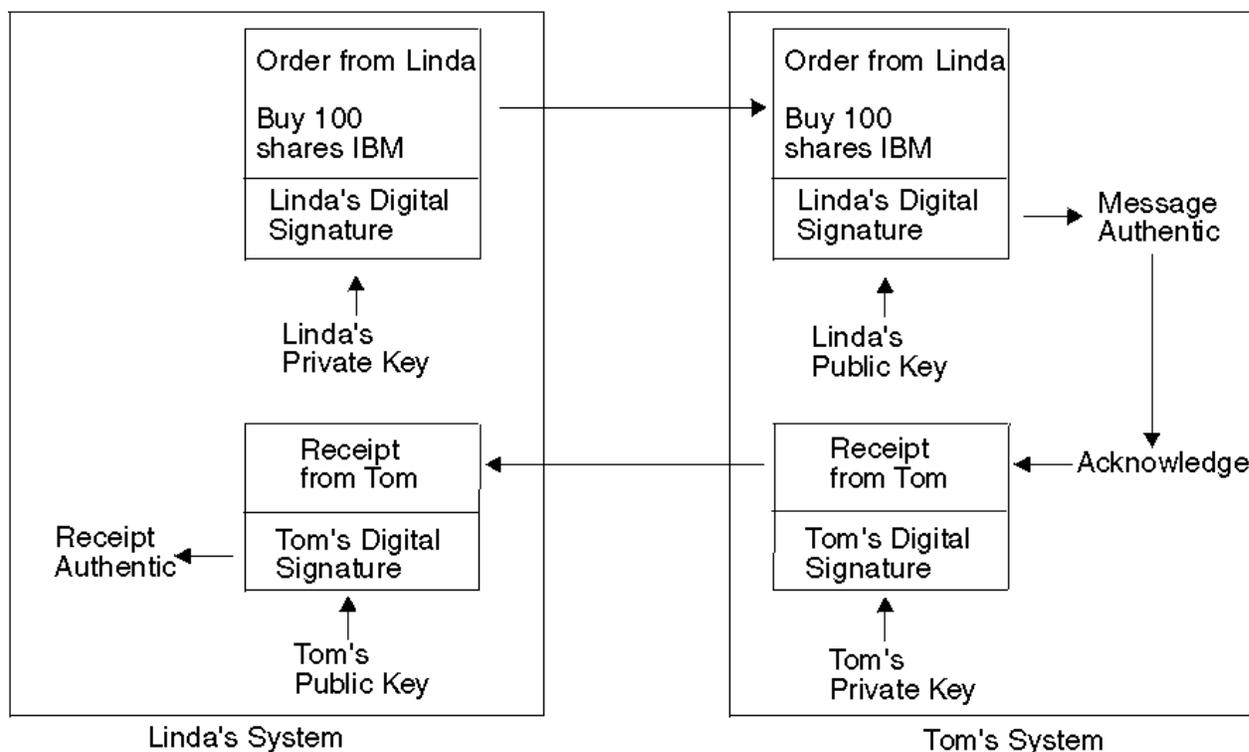


*Figure 2. An example of nonrepudiation using digital signatures*

## How does ICSF support cryptography?

ICSF supports IBM's Common Cryptographic Architecture (CCA). The CCA is based on the ANSI Data Encryption Algorithm (DEA) and the Advanced Encryption Standard (AES). DEA is also known as the U.S. National Institute of Science and Technology Data Encryption Standard (DES) algorithm. In these secret key cryptography systems, two parties share secret keys that are used to protect data and keys that are exchanged on the network. Sharing secret keys establishes a secure communications channel. The only

way to protect the security of the data in a shared secret key cryptographic system is to protect the secrecy of the secret key.

ICSF supports triple DES encryption for data privacy. Triple DES uses three, single-length keys to encipher and decipher the data. This results in a stronger form of cryptography than that available with single DES encipherment.

ICSF supports the Advanced Encryption Standard (AES). Data can be encrypted and decrypted using 128-bit, 192-bit, and 256-bit secure and clear keys. The availability of this support is the same as triple-DES.

For public key cryptography, ICSF supports the Rivest-Shamir-Adelman (RSA) algorithm, and the Elliptic Curve Digital Signature Algorithm (ECDSA). RSA, and ECDSA are the most widely used public key encryption algorithms. With public key encryption, each party establishes a pair of cryptographic keys, which includes a public key and a private key. Both parties publish their public keys in a reliable information source, and maintain their private keys in secure storage.

# How does ICSF extend the uses of cryptography?

In addition to the encryption and decryption of data, ICSF provides application programs with a callable interface to perform these tasks:

- Generate, install, and distribute DES cryptographic keys securely using both public and secret key cryptographic methods
- Generate, install, and distribute AES cryptographic keys securely using both public and secret key cryptographic methods
- Generate, verify, and translate personal identification numbers (PINs)
- Ensure the integrity of data by using message authentication codes (MACs), hashing algorithms, digital signatures, or payment card validation values.
- Develop Secure Electronic Transaction (SET) applications at the merchant and acquirer payment gateway
- PKA-encrypt and PKA-decrypt symmetric key data that Secure Sockets Layer (SSL) applications can use to generate session keys
- Develop EMV ICC applications using CSNBDKG, CSNBSPN, CSNBSKY, and CSNBPCU callable services
- Provide enhanced key management for Crypto Assist instructions
- Provide remote key loading for automated teller machines (ATMs) from a central administrative site using DES keys
- Support the EMV2000 key generation algorithm
- Enables customers to write applications implementing the Diffie-Hellman key agreement protocol using the PKA encrypt callable service (CSNDPKE)
- Provide an application programming interface (API) for applications to store objects and perform cryptographic functions using PKCS #11

## Key generation and distribution

With ICSF callable services, you can generate a variety of cryptographic keys for use on your system or distribution to other systems. You can develop key distribution protocols by using both secret key and public key cryptographic methods. With a secret key distribution system, you must first share a secret key with the system to which you intend to distribute keys. This is a major drawback with secret key distribution systems. With public key cryptography, however, you encrypt the keys you are distributing under the receiver's public key. The receiver decrypts the keys by using the receiving system's private key. Public key encryption provides methods for key distribution and authentication.

---

[1]  Invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adelman

## Personal Identification Numbers (PINs)

Many people are familiar with PINs, which enable them to use an automated teller machine (ATM). Financial networks use PINs primarily to authenticate users. Typically, the financial institution assigns a PIN. The user enters the PIN at automated teller machines (ATMs) to gain access to his or her accounts. It is extremely important to keep the PIN private, so that no one other than the account owner can use it.

ICSF enables your applications to generate PINs, to verify supplied PINs, to translate PINs from one format to another, and to store and transmit PINs in encrypted PIN blocks.

## Message Authentication Codes (MACs)

MACs are used to authenticate and verify data that is transmitted over a network, stored on the system, or stored outside the system (for example, on removable media such as tape). The MAC is generated by using the data itself and a symmetric key. The MAC is sent or stored with the data. The MAC is verified when the data is received or retrieved from storage. The MAC verification process uses the data and the symmetric key.

MACs give you these benefits:

- You can validate the authenticity of data that is transmitted over a network. You can also ensure that nothing has altered the data during transmission. For example, an active eavesdropper might tap into a transmission line, and either interject bogus messages or alter sensitive data that is being transmitted. Since the sender and the receiver share a secret key, the receiver can use a callable service to calculate a MAC on the received message. The application then compares the MAC it calculates to the MAC that was transmitted with the message. The message is accepted as genuine and unaltered only if the two MACs are identical.
- Similarly, you can store a MAC with data on tape or DASD. Then, when the system retrieves the data, an application can generate a MAC and compare it with the original MAC to detect alterations.
- In either data transmission or storage, you can use MACs in an anti-virus campaign. MACs help ensure that no unauthorized executable code has been inserted into your system.

## Hashing algorithms

The use of a hashing algorithm is another means of verifying that data has not been altered during transmission or storage. A hash, or message digest, is calculated with a public, one-way function, rather than with a secret key like a MAC. A hash, therefore, cannot be used to verify the authenticity of a message. Hashes are used in situations where it is impractical to share a secret key. For example, you can use a hash as part of a software delivery process to uncover deliberate or inadvertent modifications to software.

The originator of the data calculates the hash using the data itself and the hashing algorithm. The originator then ensures that the hash is transmitted with integrity to the intended receiver of the data. One way to ensure this is to publish the hash in a reliable source of public information. When the receiver gets the data, an application can generate a hash and compare it to the original one. If the two are equal, the data can be accepted as genuine; if they differ, the data is assumed to be bogus.

You can use the ICSF hashing algorithms to generate modification detection codes (MDCs), support the Public Key Cryptographic Standard (PKCS), and create hashes for digital signatures.

## Digital signatures

The RSA and ECC public key cryptography systems authenticate messages and their senders through the use of *digital signatures.* A digital signature on an electronically distributed document is similar to a handwritten signature on a paper document. It is not easy to forge either type of signature. Both types of signatures authenticate that the signing party either agreed to, or generated and/ agreed to, the signed document.

The originator of the data uses a hash of the data and the originator's private key to create the digital signature. The digital signature is then attached to the message. The receiver uses the originator's public key and the signed message to verify that the message was signed by the originator.

Figure 3 on page 6 is an example of using digital signatures. The sender uses a hash of the message and the private key to create the digital signature and attach it to the message before sending it to the receiving system. The receiver uses the sender's public key to regenerate the hash value from the digital signature and compares this hash value to a hash calculated on the received message. If the two hash values match, the message is considered to be authentic.
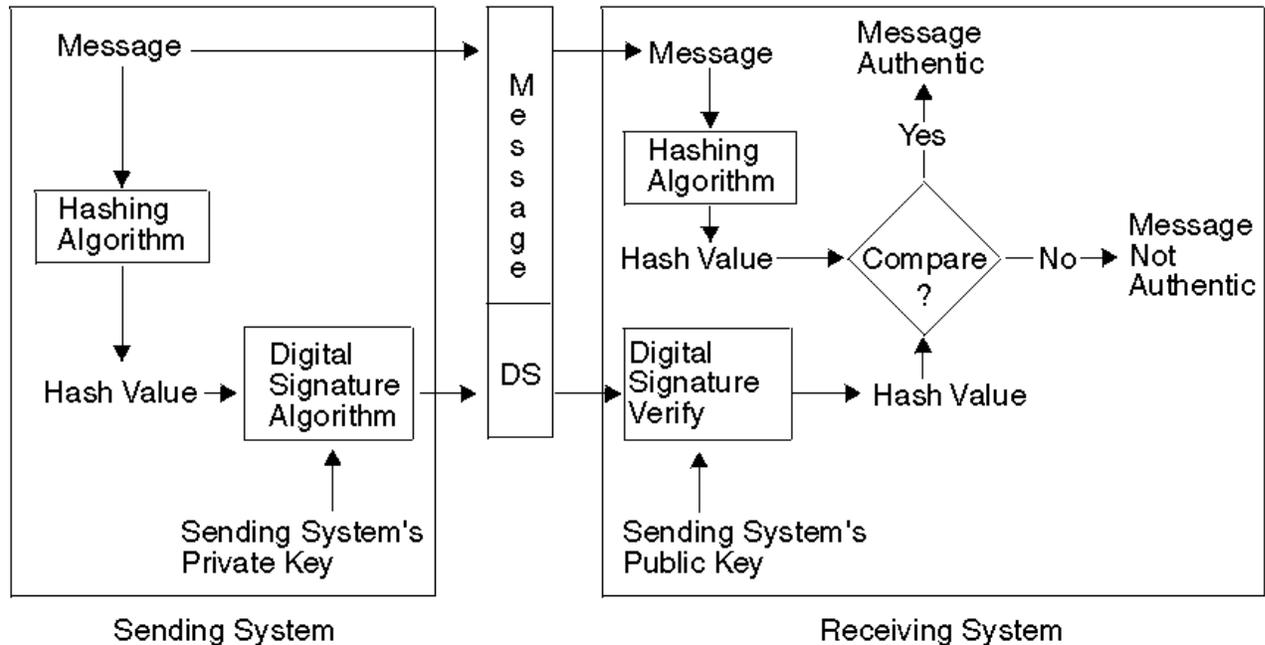


*Figure 3. Creating and verifying digital signatures in a public key system*

## Payment card verification values

Payment card standards provide for a card-validation value. ICSF supports generating and verifying these values:

- American Express card security codes (CSC)
- VISA card verification values (CVV)
- MasterCard card verification codes (CVC)
- Diner's Club card verification value (CVV)

## Translation of data and PINs in networks

Increasingly data is being transmitted across networks in which, for various reasons, the keys that are used on one network cannot be used on another network. Encrypted data and PINs that are transmitted across these boundaries must be "translated" securely from encryption under one key to encryption under another key. For example, a traveler visiting a foreign city might wish to use an ATM to access an account at home. The PIN that is entered at the ATM might need to be encrypted there and sent over one or more financial networks to the traveler's home bank. The home bank must verify the PIN before the ATM in the foreign city allows access. On intermediate systems (between networks), applications can use the Encrypted PIN translate callable service to reencrypt a PIN block from one key to another. These applications can use ICSF to ensure that PINs never appear in the clear and that the keys for encrypting the PIN are isolated on their own networks.

# SET Secure Electronic Transaction

The SET Secure Electronic Transaction standard is a global industry specification that was developed jointly by Visa International, MasterCard, and other companies. The SET protocol uses digital certificates to protect credit card transactions that are conducted over the Internet. The SET standard is a major step toward securing Internet transactions, paving the way for more merchants, financial institutions, and consumers to participate in electronic commerce.

ICSF provides callable services that support the development of SET applications that run at the merchant and acquirer payment gateway.

# Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL) is an industry-standard protocol that the Netscape Development Corporation designed to provide a data security layer between application protocols and TCP/IP. The SSL security protocol is widely deployed in applications on both the Internet and private intranets. SSL defines methods for data encryption, server authentication, message integrity, and client authentication for a TCP/IP connection. SSL uses public key and symmetric techniques to protect information.

SSL requires the decryption of a 48-byte SSL seed and the manipulation of this seed in the clear to produce symmetric session keys. The Common Cryptographic Architecture (CCA), however, does not permit even privacy session keys to appear in the clear in host storage. The ICSF SSL support services permit the RSA encryption and decryption of any PKCS 1.2-formatted symmetric key data. The PKA encrypt callable service CSNDPKE encrypts a supplied clear key under an RSA public key. Using the PKA decrypt callable service CSNDPKD makes it possible to unwrap the RSA-encrypted SSL seed and return it to the application in the clear. The application can then use this clear key to generate session encryption keys.

# EMV integrated circuit card specifications

EMV stands for Europay, MasterCard, and Visa, the three companies that originally created the common standard for retail terminals accepting chip cards. Chip cards are also called stored value cards or smart cards. An algorithm or formula is stored in the chip. Chip card transactions are PIN-based for maximum security.

In addition to the EMV specification for contact chip cards, there are also specifications for contactless chip cards, common payment applications (CPA), card personalization, mobile payments, and tokenisation. These other specifications are being used by mobile payment systems and e-wallets.

The EMV standard is now managed by EMVCo, a consortium with control split equally among American Express, China UnionPay, Discover, Japan Credit Bureau (JCB), Mastercard, and Visa. EMVCo also has a variety of associates that include retailers, banks, payment processors, and other credit card companies and financial institutions. These associates provide both technical and strategic business input to EMVCo.

With ICSF, you can develop EMV ICC integrated circuit card applications using diversified key generate (CSNBDKG), secure messaging for PINs (CSNBSPN) and secure messaging for keys (CSNBSKY) services. ICSF supports the PIN change algorithms specified in the VISA Integrated Circuit Card Specification. PIN block/change (CSNBPCU), provides this support. Additionally, the diversified key generate service (CSNBDKG) supports the EMV2000 key generation algorithm.

ICSF callable services also simplify payment processing. These services use parameters that are specifically for EMV functions and call the correct sequence of existing ICSF services and cryptographic coprocessor services to perform EMV functions.

**Generate Issuer Master Key Service (CSNBGIM)**
This service helps with the initial steps of EMV setup by generating and storing the issuer master keys. The master keys are returned in either internal or external key tokens for key management.

**Derive ICC Master Key Service (CSNBDCM)**
This service generates an ICC master key from an issuer master key. The ICC master keys are needed for ICC personalization, EMV transaction processing, and EMV scripting. The master keys are returned in either internal or external key tokens for key management.

**Derive Session Key Service (CSNBDSK)**
> This service generates a session key from either an issuer master key or an ICC master key. Session keys are needed for EMV transaction processing and EMV scripting.

**EMV Transaction (ARQC/ARPC) Service (CSNBEAC)**
> This service simplifies EMV ARQC and ARPC transaction processing.

**EMV Scripting Service (CSNBESC)**
> This service simplifies EMV scripting. Scripts may be encrypted for confidentiality, MAC'd for integrity, or both.

**EMV Verification Service (CSNBEVF)**
> This service provides additional functions used by MasterCard:
>
> - Verification of data authentication codes.
> - Verification of ICC dynamic numbers.
> - Decryption of encrypted counters.

# ATM remote key loading

The process of remote key loading is loading DES keys to automated teller machines (ATMs) from a central administrative site. Because a new ATM has none of the bank's keys installed, getting the first key securely loaded is currently done manually by loading the first key-encrypting key (KEK) in multiple cleartext key parts. ANSI X9.24-2 defines the acceptable methods of doing this using public key cryptographic techniques, which will allow banks to load the initial KEKs without having to send anyone to the ATMs. This method is quicker, more reliable and much less expensive.

Once an ATM is in operation, the bank can install new keys as needed by sending them enciphered under a KEK it installed at an earlier time. Cryptographic architecture in the ATMs is not Common Cryptographic Architecture (CCA) and it is difficult to export CCA keys in a form understood by the ATM. Remote key loading makes it easier to export keys to non-CCA systems without compromising security.

The ANSI TR-34 protocol allows keys to be remotely loaded into ATMs and other devices. TR-34 uses asymmetric techniques to distribute symmetric keys between two devices that share asymmetric keys. This method is designed to operate within the existing capabilities of devices used in the retail financial services industry. It is an implementation of the Unilateral Key Transport Method defined in ANS X9.24-2.

# Public Key Cryptography Standard #11 (PKCS #11)

PKCS #11 specifies an application programming interface (API) to devices (virtual or real) which hold cryptographic information and perform cryptographic functions. Applications written in C can code to the PKCS #11 cryptographic API and on the z/OS platform ICSF will be invoked in order to manage PKCS #11 tokens and objects and to perform cryptographic functions.

PKCS #11 supports Java Security's use of the PKCS #11 API and allows Java applications, RACF and SSL to replace their individual key stores with a single repository for keys managed by ICSF.

For more information about the specific PKCS #11 APIs see *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*.

# DK AES PIN support

The German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) designed methods of creating, processing, and verifying PINs for its members. The methods use a PIN reference value (PRW) that is generated when a PIN is created or changed and used to verify the PIN supplied in a transaction. The methods are not dependent on a specific cryptographic algorithm, but DK has chosen the AES algorithm for its implementation.

# Chapter 2. Solving your business needs with ICSF

As more businesses automate their operations and start conducting electronic commerce over the Internet, the increased use of workstations and automated teller machines generates high transaction loads. Attacks on security are becoming more sophisticated. Criminals can gain tremendous payoffs from wiretapping and theft of data from storage.

Electronic commerce, electronic funds transfer (EFT), and electronic data interchange (EDI) applications can use ICSF callable services to secure Internet transactions. These applications can make use of cryptography to protect funds transfers, purchase orders, letters of intent, contracts, credit card information, and other sensitive data from the risks of theft, fraud, or sabotage. A business can also decrease the amount of sensitive material that is exchanged by couriers. This allows a business to provide better service, become more competitive, and potentially reduce its expenses.

ICSF provides a high level of security and integrity for your data. It can help you meet many of the current needs and the future needs of your business by solving many of the information system security problems you face. This topic explains how you can use ICSF for data security, key exchange, and personal authentication.

## Keeping your data private

ICSF cryptographic functions are specifically designed for high security. In addition, ICSF also provides these security precautions:

- ICSF uses the DES and AES algorithms, which are widely regarded as highly secure, to encipher and decipher data.
- The master keys are stored in highly secure hardware.
- DES keys, AES keys and PKA private keys may be encrypted under a master key for protection.
- You can use cryptographic keys only for their intended function. For example, a program that uses a key to verify a MAC cannot use the same key to generate MACs.
- You can use IBM Resource Access Control Facility (RACF) to control access to specific ICSF callable services, to specific keys that are stored in a CKDS, PKDS or TKDS or to both. RACF can also be used to protect the use of tokens passed in when calling a service using the Key Store Policy.
- With the optional Trusted Key Entry (TKE) workstation, you can create a logical secure channel. You can then use this channel to distribute master keys and operational keys to remote systems. The TKE workstation is particularly suited to the distributed computing environment that requires remote key management of one or more systems. For added security, you can require that multiple security officers perform critical operations or you can implement TKE smart card support.

## Transporting data securely across a network

You may need to protect data that is sent between two applications when the data must pass through one or more intermediate systems.

In a DES cryptographic system, if the two applications cannot share a key, you must set up an application on one or more of the intermediate systems to translate the ciphertext from encryption under the sending system's key. Translation re-encrypts the ciphertext under a new key for which the receiving system has a complementary key.

An application can use the ICSF ciphertext translate callable service to do this. ICSF prevents the recovery of plaintext on intermediate systems, because you cannot decrypt the data with the same key that is used to translate the ciphertext on the intermediate system. illustrates the use of the ciphertext translate callable service.

**9**
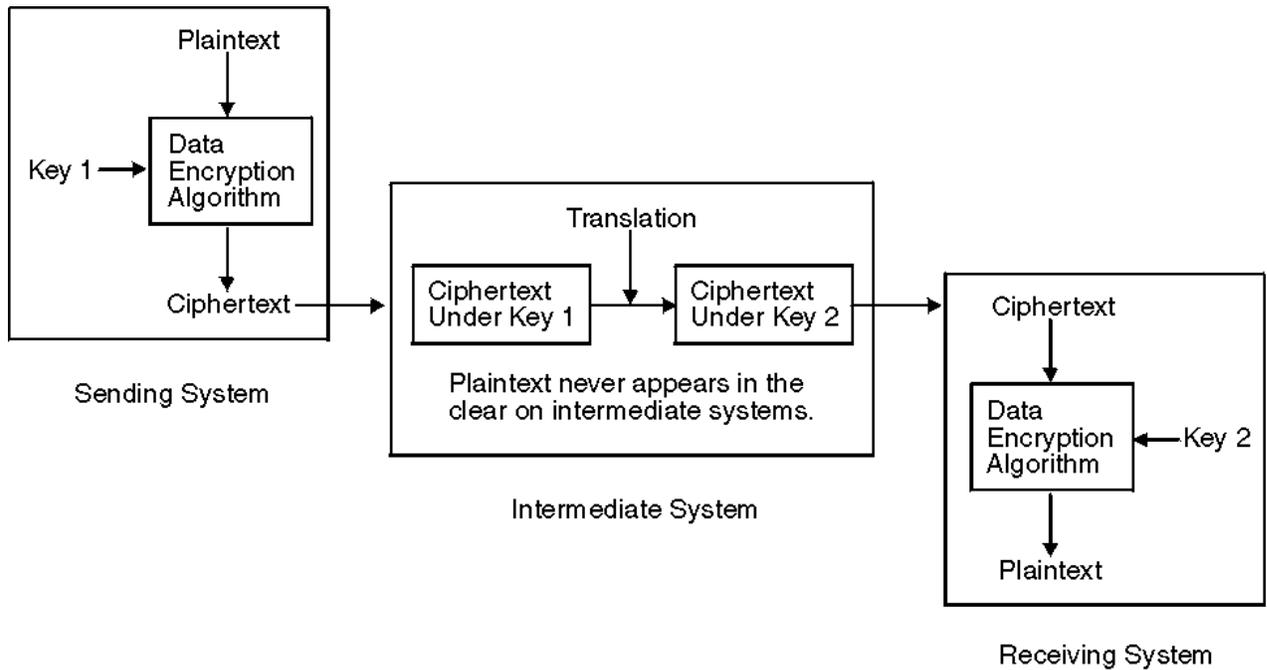
*Figure 4. DES encrypted data protected when sent on intermediate systems*

In a PKA cryptographic system, you can develop an application that does not require translation of ciphertext by the intermediate systems. The sender enciphers the message by using a DES or AES data-encrypting key. The sender then uses the receiver's PKA public key to encipher the DES or AES data-encrypting key. The intermediate system merely transfers the ciphertext and the enciphered key to the receiving system. The intermediate system does not have the receiver's PKA private key and, therefore, cannot decipher the enciphered data-encrypting key. Without the deciphered data-encrypting key, the intermediate system cannot decipher the message. The receiving system uses its PKA private key to decipher the DES or AES data-encrypting key, which it then uses to decipher the message .
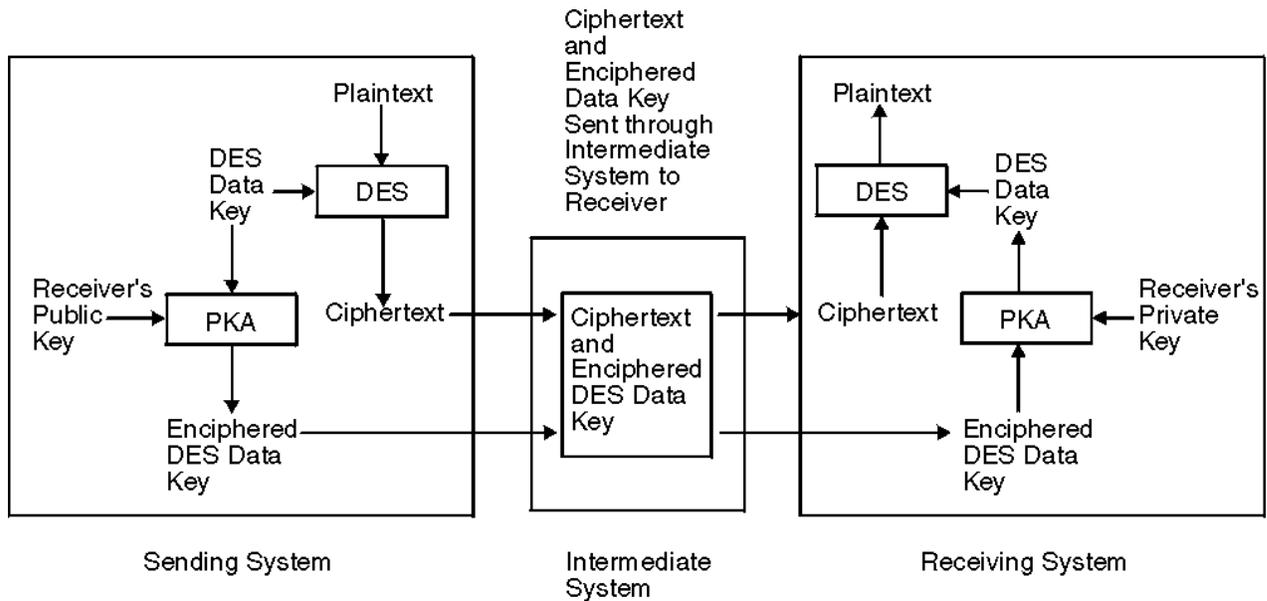


*Figure 5. PKA encrypted data protected when sent on intermediate systems*

## Supporting the Internet Secure Sockets Layer protocol

The Secure Sockets Layer (SSL) provides a data security layer between the network layer and various internet transfer protocol applications. For example, SSL can provide a secure session between the transmission control protocol/internet protocol (TCP/IP) network layer and the hypertext transfer protocol (HTTP) or file transfer protocol (FTP) application. SSL provides data encryption, message integrity, and server authentication for TCP/IP connections between clients and servers. SSL ensures that credit card numbers and other sensitive information can be sent over the Internet without fear of interception.

To begin a secure session, the server and client exchange a handshake. In this digital handshake, the client and server are authenticated and also agree on the SSL version, data compression method, and cryptographic algorithm they will use when exchanging data. They also exchange an RSA-encrypted seed key that SSL manipulates to create symmetric session keys that are used to encrypt the data that the client and server exchange. The ICSF PKA encrypt and PKA decrypt callable services provide a secure method for SSL applications to exchange this seed key.

You can exploit Crypto Express accelerators without entering master keys if SSL uses clear keys. This enhances performance.

# Transacting commerce on the Internet

The Internet is rapidly becoming a major arena of commerce. For electronic commerce to grow to its full potential, however, we need to resolve several barriers to buying and selling over the Internet. Consumers are reluctant to send their bank card data over the Internet without assurances that this information is secure. Merchants need to be able to determine the clear identities of their online customers. The SET Secure Electronic Transaction protocol can help to break down these major barriers to electronic commerce. MasterCard and Visa, with the assistance of IBM and a number of technology industry partners, cooperatively developed the SET protocol.

SET is an industry-wide, open standard for online credit card transactions. The SET protocol addresses the transaction payment phase of a transaction from the individual, to the merchant, to the acquirer (the merchant's current credit card processor). The SET protocol ensures the privacy and integrity of real time bank card payments over the Internet. In addition, with SET in place, everyone in the payment process knows the identity of everyone else. The core protocol of SET is the use of digital certificates to fully authenticate the card holder, the merchant, and the acquirer. Each participant in the payment transaction holds a certificate that validates his or her identity. Public key cryptography makes it possible to exchange, check, and validate these digital certificates for every Internet transaction. The mechanics of this operation are transparent to the application.

Under the SET protocol, a digital certificate which identifies the card-holder to the merchant must accompany every online purchase. The buyer's digital certificate serves as an electronic representation of the buyer's bank card but does not actually show the credit card number to the merchant. The merchant's SET application authenticates the buyer's identity. The application then decrypts the order information, processes the order, and forwards the still-encrypted payment information to the acquirer for processing. The acquirer's SET application authenticates the buyer's credit card information, identifies the merchant, and arranges settlement. With SET, the Internet becomes a safer, more secure environment for the use of payment cards.

# Exchanging keys safely between networks

The practice of transmitting clear keys between networks can be a security exposure. Persons that obtain the clear keys can use them to decrypt transmitted data. ICSF offers several ways to eliminate this problem and ensure that keys are transmitted safely.

## Exchanging symmetric keys using callable services

ICSF provides these security measures for AES and DES key exchange:

- Encrypting the keys to be sent between systems, so that they are not in the clear.

- Requiring that specialized transport keys protect the data-encrypting keys or key-encrypting keys. Transport keys can be used only to protect other keys; they cannot be used for other cryptographic operations.
- Requiring that the sending (exporting) and receiving (importing) of a key be by two different, complementary forms of the same transport key (for example, export and import). These two forms are complements of each other. You cannot use a key in place of its complement.
- Requiring that a key protected under a transport key be made no longer operational—that is, not usable for other cryptographic functions such as encryption, MAC verification, and PIN verification. Only the receiving system can make a protected key operational.

An "exported" key is a key that leaves your system. The transport key that is used to protect it is called an exporter key-encrypting key. When another system receives the key, the key is still protected under the same key-encrypting key. This key-encrypting key must be installed as an importer key-encrypting key on the receiving system. Before two systems can exchange keys, they must establish pairs of transport keys. The exporter key-encrypting key and the importer key-encrypting key are a complementary pair. You can set up pairs of transport keys, using the key generator utility program (KGUP) or callable services. To exchange keys in only one direction, you need a single pair of transport keys. To exchange keys in both directions, you need two pairs of transport keys. The illustration in shows an example of using DES transport keys to exchange keys between systems.
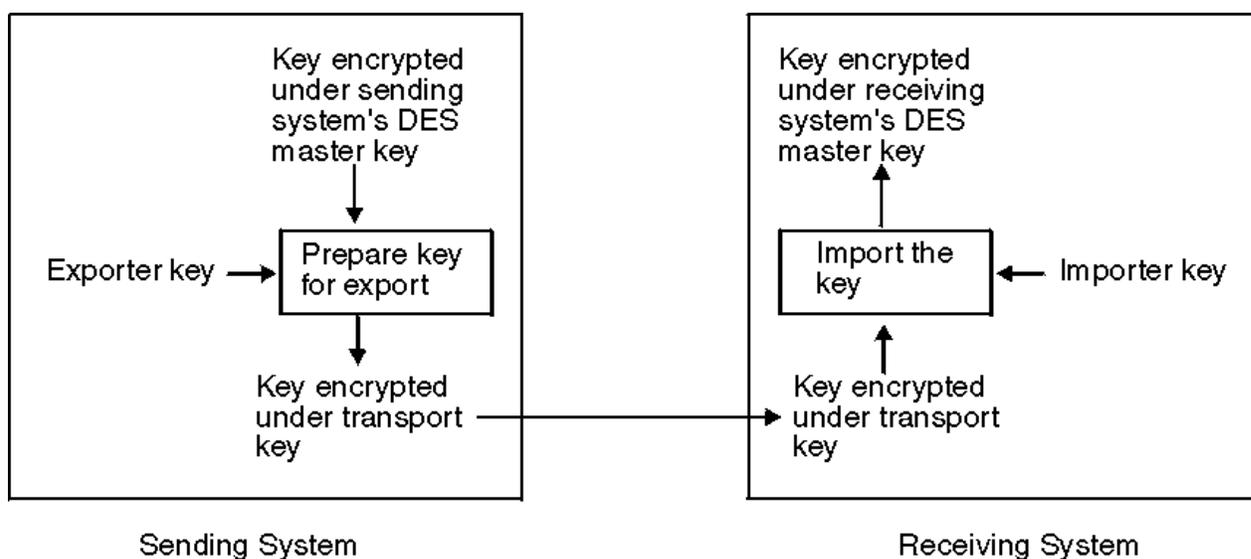


*Figure 6. Key exchange in a DES cryptographic system*

**Note:** In Program Cryptographic Facility (PCF) applications, transport keys could only protect data-encrypting keys. In ICSF, all DES keys can be protected and securely distributed through the use of transport keys.

## Exchanging DES or AES data-encrypting keys using an RSA key scheme

The ability to create secure key-exchange systems is one of the advantages of combining DES or AES and PKA support in the same cryptographic system. Because PKA cryptography uses more intensive computations than DES or AES cryptography, it is not the method of choice for all cryptographic functions. PKA cryptography enhances the security of DES or AES key exchanges. DES or AES data-encrypting keys that are encrypted using an RSA public key can be exchanged safely between two systems. The sending system and the receiving system do not need to share a secret key to be able to exchange RSA-encrypted DES or AES data-encrypting keys. shows an example of this. The sending system enciphers the DES data-encrypting key under the receiver's RSA public key and sends the enciphered data-encrypting key to the receiver. The receiver deciphers the data-encrypting key by using the receiving system's RSA private key.
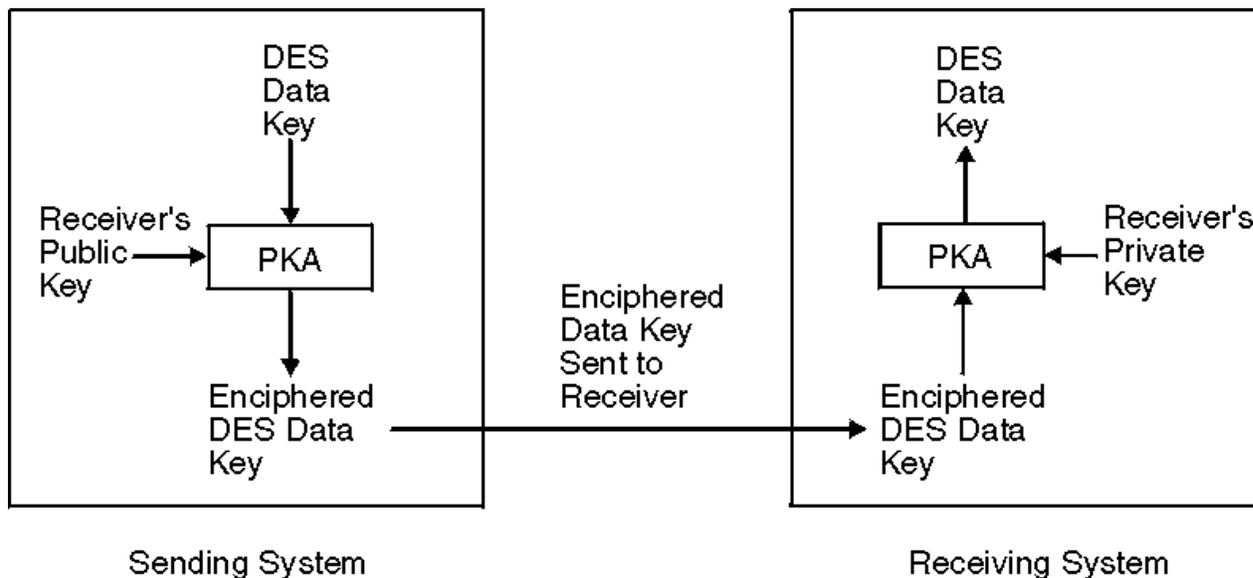
*Figure 7. Distributing a DES data-encrypting key using an RSA cryptographic scheme*

## Creating DES or AES Keys using an ECC Diffie-Hellman key scheme

ECC Diffie-Hellman allows two systems to create a symmetric key without exchanging the key.

The sender's private ECC key and the receiver's public ECC keys are combined with the party information to generate a symmetric key that could be used to encipher a message. The ciphertext is transferred to the receiver's system. On the receiver's system, the receiver's private ECC key and the sender's public keys are combined with the party information to generate a symmetric key that could be used to decrypt the ciphertext.

## Exchanging keys and their attributes with non-CCA systems

A TR-31 key block is a format defined by the American National Standards Institute (ANSI) to support the interchange of keys in a secure manner with key attributes included in the exchanged data. The TR-31 key block format has a set of defined key attributes that are securely bound to the key so that they can be transported together between any two systems that both understand the TR-31 format. ICSF enables applications to convert a CCA token to a TR-31 key block for export to another party, and to convert an imported TR-31 key block to a CCA token. This enables you to securely exchange keys and their attributes with non-CCA systems.

DES/TDES and AES keys can be transported in TR-31 key blocks.

Refer to *z/OS Cryptographic Services ICSF Application Programmer's Guide* for more information.

## Managing master keys using a Trusted Key Entry workstation

ICSF supports the Trusted Key Entry (TKE) workstation. It is available as an optional feature on all IBM eServer, IBM Z®, and IBM zEnterprise servers.

The TKE workstation enables the creation of a logically secure channel for master key entry and key distribution. All versions of the TKE workstation are secure.

### Integrity and Privacy

The TKE workstation uses a variety of public key cryptographic techniques to ensure both the integrity and the privacy of the master key transfer channel. In addition, you can use a single TKE workstation to set up master keys in all the cryptographic coprocessors to which it is TCP/IP attached without manual intervention. The TKE workstation also provides support for loading operational keys on systems with cryptographic coprocessors.

# Using Personal Identification Numbers (PINs) for personal authentication

*Personal authentication* is the process of validating personal identities. The personal identification number (PIN) is the basis for verifying the identity of a customer across financial industry networks. ICSF provides callable services to generate and verify PINs, and translate PIN blocks. You can use the callable services to prevent unauthorized disclosures when organizations handle PINs. Except for the Clear PIN generate callable service, PINs never appear in the clear.

ICSF provides services for handling a wide variety of PIN block formats, including:

- ISO Format 0 (same as ANSI X9.8, ECI Format 1, and VISA Format 1)
- ISO Format 1 (same as ECI Format 4)
- ISO Format 2
- ISO Format 3
- ISO Format 4
- VISA Format 2
- VISA Format 3
- VISA Format 4
- IBM 4704 Encrypting PINPAD Format
- IBM 3624 Format
- IBM 3621 Format (same as IBM 5906)
- ECI Format 2
- ECI Format 3


ICSF also supports these Clear PIN generate and verification algorithms:

- IBM 3624 Institution-Assigned PIN
- IBM 3624 Customer-Selected PIN (through a PIN offset)
- VISA PIN (through a VISA PIN validation value)
- Interbank PIN

For more information about PIN block formats and the ICSF callable services that support PINs, refer to 'Financial Services' in *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

# Verifying data integrity and authenticity

ICSF provides several processes for verifying the integrity of transmitted messages and stored data:

- Message authentication codes (MAC)
- Modification detection codes (MDC) or hashes
- Digital signatures
- VISA card-verification value, MasterCard Card Verification Code, Diner's Club CVV, American Express card security codes

These processes enable your applications to verify that a message you have received has not been altered. The message itself can be in clear or encrypted form. In addition, digital signatures also authenticate the message sender's identity. VISA card-verification values ensure the safe transmission of credit card information over a computer network.

Your choice of callable service depends on the security requirements of your environment. If the sender and receiver share a secret key, use MAC processing to ensure both the authenticity of the sender and the integrity of the data. If the sender and receiver do not share a secret key, use a digital signature to ensure

both the authenticity of the sender and the integrity of the data. If the sender and the receiver do not share a secret cryptographic key and you need to ensure only the integrity of transmitted data, use a hashing process.

## Using Message Authentication Codes

To use message authentication when sending a message, an application generates a MAC for it using the MAC generate callable service and one of these methods:

- The ANSI standard X9.9, option 1 with a DES MAC or DATA key.
- The X9.19 optional double key MAC procedure with a DES MAC key.
- The EMV padding rules with a DES MAC key.
- The ISO 16609 CBC mode with a DES MAC or DATA key.
- The AES XCBC MAC algorithm from the IETF RFC 3566 which uses AES DATA key to produce a 96-bit result.
- The AES XCBC PRF algorithm from the IETF RFC 3566 which uses AES DATA key to produce a 128-bit result.
- The FIPS-198 Keyed-Hash Message Authentication Code (HMAC) algorithm with a variable length HMAC key.
- The NIST SP 800-38B cipher message authentication code (CMAC) algorithm with AES and DES MAC keys.

The originator of the message then sends the MAC with the message text.

When the receiver gets the message, an application program calls the MAC verification callable service. The service again encrypts the message text by using the same method that was used to compute the original MAC. The callable service then notifies the receiver whether the MAC has been verified or not. The callable service does not allow the receiver to have access to the MAC it generates. Because the sender and the receiver share secret cryptographic keys that are used in the MAC calculation, the MAC comparison also ensures the authenticity of the message.

## Generating and verifying digital signatures

An application generates a digital signature for a message by first supplying a hash of the message to the digital signature generate callable service. The callable service then uses the signer's private key to create the signature. ICSF supports the use of RSA and ECC digital signatures. To verify the digital signature, the receiver's application supplies a hash of the message and the digital signature to the digital signature verify callable service. The callable service then uses the sender's public key to verify the signature. A return code indicates that the verification either succeeded or failed. Figure 3 on page 6 provides an example of using digital signatures.

## Using modification detection codes and message hashing

When you are sending a message, use either the MDC generate callable service, or the one-way hash generate callable service to generate a message hash. The choice depends on the cryptographic standard you are using.

The MDC is a 128-bit value that is generated by a one-way cryptographic calculation. The originator of the message transmits the MDC with integrity to the intended receiver of the file. For instance, the originator could publish the MDC in a reliable source of public information. The receiver of the message can use an application program and the same callable service to generate another MDC. If the two MDCs are identical, the receiver assumes that the message is genuine. If they differ, the receiver assumes that someone or some event altered the message.

A hash is a message digest that is generated by a one-way cryptographic calculation. ICSF supports these hash algorithms:

- MD5 produces a 128-bit hash value

- SHA-1 produces a 160-bit hash value
- SHA-224 produces a 224-bit hash value
- SHA-256 produces a 256-bit hash value
- SHA-384 produces a 384-bit hash value
- SHA-512 produces a 512-bit hash value
- RIPEMD-160 produces a 160-bit hash value

Applications can use the hash value and the originator's private key to generate a digital signature and attach it to the message. The receiver of the message uses the originator's public key to authenticate the digital signature.

Both MACs and hashes can be used similarly to ensure the integrity of data that is stored on the system or on removable media such as tape.

## Verifying payment card data

The Visa International Service Association (VISA) and MasterCard International, Incorporated have specified a cryptographic method to calculate the VISA card-verification value (CVV) and the MasterCard card-verification code (CVC). This value relates to the personal account number (PAN), the card expiration date, and the service code and is used to detect forged cards. The CVC can be encoded on either track 1 or track 2 of a magnetic-striped card. Because most online transactions use track-2, the ICSF callable services generate and verify the CVV[2] by the track-2 method.

The VISA CVV service generate callable service calculates a 1- to 5-byte CVV. This value results from using two data-encrypting keys to DES-encrypt the PAN, the card expiration date, and the service code. The VISA CVV service verify callable service calculates the CVV by the same method. The service compares the CVV it calculates to the CVV supplied by the application (which reads the credit card's magnetic stripe). The service then issues a return code that indicates whether the card is authentic.

The Transaction Validation callable service can be used to generate and verify American Express card security codes (CSC). The service supports 3, 4, and 5 character codes and versions 1.0 and 2.0 algorithms.

## Maintaining continuous operations

ICSF provides continuous cryptographic operations. Cryptographic keys stored in a cryptographic key data set (CKDS or PKDS) can be reenciphered under a new master key or updated by using either the key generator utility program or the dynamic CKDS or PKDS update callable services. ICSF performs these updates without disrupting applications in process. With PCF, you need to stop cryptographic functions before changing the master key or updating the CKDS or PKDS. You do not need to stop ICSF or interrupt cryptographic applications before changing the master keys, refreshing the CKDS or PKDS, or dynamically updating either the CKDS or PKDS.

**Note:** The ability to change the master keys or update the CKDS or PKDS without interruption requires that ICSF be running in noncompatibility mode. That is, you must convert all existing PCF applications to the new callable services. For a description of noncompatibility mode, see .

These features and actions enhance the security of cryptographic functions:

- Performing cryptographic calculations and storing master keys within tamper-resistant hardware
- Enforcing separation of DES and AES keys
- Controlling access to functions and keys through the use of RACF
- Generating system management facility (SMF) audit records

---

[2] The VISA CVV and the MasterCard CVC refer to the same value. This information uses CVV to mean both CVV and CVC.

ICSF supports a Geographically Dispersed Parallel Sysplex (GDPS) environment by allowing KDS updates on a production system or sysplex to be propagated to another production system or sysplex or to a backup or Disaster Recovery (DR) system or sysplex. For more information about configuring ICSF in a GDPS environment, see *z/OS Cryptographic Services ICSF Administrator's Guide*.

# Dynamic service update

Dynamic service update allows you to apply service updates with minimal impact to ICSF availability. ICSF can activate service without a manual stop and start of ICSF. These updates include service updates as well as changes to the options data set that cannot be applied via the SETICSF OPT,REFRESH command. Additionally, dynamic service updates can be used to recycle ICSF when there are problems that are not resolving.

Before starting a dynamic service update, see 'Dynamic service update' in *z/OS Cryptographic Services ICSF System Programmer's Guide*.

# Reducing costs by improving productivity

ICSF improves productivity by simplifying routine operations and providing interfaces and callable services that help you manage your enterprise's cryptographic environment.

ICSF simplifies the job of the security administrator by providing ISPF dialogs for key management and distribution. ICSF also provides a pass phrase initialization procedure that generates and loads all needed master keys. Use pass phrase initialization to fully enable your cryptographic system in a minimum of steps. In addition, a series of Master Key Entry panels simplifies the master key entry procedure. These panels permit the administrator to change the master keys without interrupting application programs that use cryptographic functions.

In enterprises that require enhanced key-entry security, a Trusted Key Entry (TKE) workstation is available as an optional feature. The TKE workstation allows the security administrator to securely load master keys and operational keys. The security administrator can use the TKE workstation to load keys into multiple cryptographic hardware features from a remote location.

ICSF provides the application programmer with a set of callable services that support cryptographic functions and key management protocols. Applications written in Assembler and several high-level programming languages can use these callable services.

ICSF provides the systems programmer with an easy method of setting and changing the ICSF installation options. The systems programmer needs only to edit an options data set rather than altering an object module. ICSF provides a sample installation options data set in members CSFPRM00 of SYS1.SAMPLIB.

# Improving cryptographic performance

ICSF uses state-of-the-art hardware to improve performance of DES, AES, and PKA calculations. Both cryptographic coprocessor and cryptographic accelerator hardware is used by ICSF. The CEX2C, CEX3C, CEX4C, CEX5C, CEX6C, and CEX7C coprocessors and CEX2A, CEX3A, CEX4A, CEX5A, CEX6A, and CEX7A accelerators are referred as Cryptographic Features. This can remove limitations on the growth of your installation and enable it to use cryptography in high-transaction-rate applications. ICSF also improves performance by exploiting z/OS and by using an in-storage copy of the CKDS and PKDS. Maintaining DES, AES or PKA cryptographic keys in a protected data space (in addition to a data set) improves performance and availability by reducing requirements for read access to cryptographic keys.

## Using RMF and SMF to monitor z/OS ICSF events

You can run ICSF in different configurations and use installation options to affect ICSF performance. While ICSF is running, you can use the Resource Measurement Facilities (RMF) and System Management Facilities (SMF) to monitor certain events. For example, ICSF records information in the MVS SMF data set when ICSF status changes in a processor or when you enter or change the master key. ICSF also sends information and diagnostic messages to data sets and consoles.

With the availability of cryptographic hardware on an LPAR basis, RMF provides performance monitoring in the Postprocessor Crypto Hardware Activity report. This report is based on SMF record type 70, subtype 2. The Monitor I gathering options on the REPORTS control statement are CRYPTO and NOCRYPTO. Specify CRYPTO to measure cryptographic hardware activity and NOCRYPTO to suppress the gathering. In addition, overview criteria is shown for the Postprocessor in the Postprocessor Workload Activity Report - Goal Mode (WLMGL) report. For more information, see *z/OS RMF Programmer's Guide*, *z/OS RMF User's Guide*, and *z/OS RMF Report Analysis*.

ICSF also supports enabling RMF to provide performance measurements on ICSF services (Encipher, Decipher, MAC Generate, MAC Verify, One Way Hash, PIN Translate, PIN Verify, Digital Signature Generate, and Digital Signature Verify).

These functions are also performed on the cryptographic feature coprocessors, except for One-Way Hash, which runs on the CPACF (CP Assist for Cryptographic Functions). ICSF counts the number of requests to the cryptographic processors as the instruction counts in the measurements.

ICSF also supports cryptographic usage tracking by using SMF record type 82 subtype 31 records. These records indicate the job name, user ID, and count associated with cryptographic usage. For more information, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

For diagnosis monitoring, use Interactive Problem Control System (IPCS) to access the trace buffer and to format control blocks.

## Improving performance in a CICS environment

ICSF supports a CICS-ICSF attachment facility that improves the performance of applications in the CICS regions when an application in the region requests a long-running ICSF service. The attachment facility consists, in part, of a CICS Task Related User Exit (TRUE) that attaches a task control block that does the actual call to the ICSF services. The CICS Resource Manager Interface allows a CICS application program to invoke code that is not written expressly for use under CICS, using the application programming interface that is native to that code. Code that is accessed in this manner is called a resource manager. In the case of the CICS-ICSF attachment facility, ICSF becomes a resource manager for CICS. This means that a CICS application desiring to use long-running ICSF services (such as PKA operations) can be placed in a CICS WAIT rather than an OS WAIT for the duration of the operation. This results in improved performance for other applications that are running in the same CICS region.

The CICS TRUE off loads CICS transaction cryptographic work that might give up control to a z/OS subtask. Synchronous work done on the CCFs would not benefit from the use of the TRUE. This is quite different when a cryptographic feature is used. All work that is performed on these features is asynchronous, and gives up control at least once due to PAUSE processing or LATCH suspension. If the TRUE is not used under CICS, cryptographic work directed to these features will be effectively single threaded. Use of the CICS TRUE is mandatory. The default CICS WAITLIST contains the names of all services that use the cryptographic features and should be used without modification.

For additional information about installing the CICS-ICSF attachment facility or creating a modifiable CICS Wait List, refer to the WAITLIST parameter in *z/OS Cryptographic Services ICSF System Programmer's Guide*. The parameter is an option in the Installation Options data set and points to a modifiable data set which contains the names of services that are placed in the CICS Wait List. If this option is not specified, the default ICSF CICS Wait List will be utilized by ICSF when a CICS application invokes an ICSF callable service.

## Customizing ICSF to meet your installation's needs

ICSF provides the flexibility your installation needs to customize your cryptographic system.

## Using ICSF exits to meet special needs

Exits are programs that your system programmer writes to meet your installation's particular needs. These exits (and installation-defined callable services) perform tasks such as tailoring, monitoring, changing, or diagnosing ICSF. Use of such interfaces can create dependencies on the detailed design or implementation of ICSF. For this reason, use installation exits only for these specialized purposes.

ICSF exits include:

- Exits called when an operator command starts, stops, or changes ICSF
- An exit for each of the callable services
- Exits that are called when you access the disk copy of the CKDS
- An exit that is called when an application accesses the in-storage CKDS

For more information about ICSF exits, refer to 'Installation Exits' in *z/OS Cryptographic Services ICSF System Programmer's Guide*.

## Creating installation-defined callable services

Your installation can define a callable service that will run in the ICSF address space and have access to selected ICSF control blocks.

The UDX function is invoked by an "installation-defined" or generic callable service. The callable service is defined in the Installation Options data set (UDX parameter) and the service stub is link-edited with the application. The application program calls the service stub which accesses the UDX installation-defined service.

There is a one-to-one correspondence between a specific generic service in ICSF and a specific UDX command processor in the cryptographic coprocessor. UDXs are authorized using the TKE workstation. Authorization is not LPAR specific. See 'Managing User Defined Extensions' in *z/OS Cryptographic Services ICSF Administrator's Guide* for additional information. Contact IBM Global Services for any problems with UDX.

Development of a UDX for a cryptographic coprocessor requires a special contract with IBM.

## Using options to tailor ICSF

ICSF lets your installation use different sets of options at different times in the operation of your system. Your installation can specify which options are in each set. These are some of your choices:

- You can choose which of three migration options to use when migrating from or coexisting with PCF: noncompatibility mode, compatibility mode, or coexistence mode.

  For more information on running PCF applications with ICSF, refer to "Running PCF applications under ICSF" on page 39.


- You can allow processing in special secure mode, in which you can work with clear keys and clear PINs. Alternatively, you can disallow processing in that mode.
- For each exit point, you can specify the name of the exit routine and operating information.
- You can alter the REASONCODES options parameter in the Installation Options data set to determine which set of reason codes (ICSF or TSS values) are returned to application service calls. If the REASONCODES option is not specified, the default of REASONCODES(ICSF) is used. The codes will only be converted if there is a 1-to-1 correspondence.
- You can use the WAITLIST (data_set_name) options parameter in the Installation Options Data Set to point to a modifiable data set that contains the names of services that are placed into the CICS Wait List. If the WAITLIST option is not specified, the default ICSF CICS Wait List will be utilized by ICSF when a CICS application invokes an ICSF callable service.
- You can use the UDX(UDX-id,service-number,load-module name,'comment_text',FAIL(fail-option)) parameter to define a User Defined Extension (UDX) service to ICSF.

## Isolating and protecting PR/SM partitions

If you are using the Processor Resource/Systems Manager (PR/SM) feature to run in logically partitioned mode, each PR/SM partition is able to use its own master keys on the cryptographic features. This allows

your installation to have multiple independent cryptographic systems running on the same processor with the same degree of isolation and protection as if they were running on physically separate processors.

# Enabling growth

For applications that need to protect critical data against disclosure or modification, ICSF provides callable services that enable high-level language applications to easily access the system's underlying cryptographic functions.

By providing callable services that comply with IBM's Common Cryptographic Architecture (CCA), ICSF enables application designers and programmers to extend the uses of their current applications. Most of the callable services provided by ICSF are also provided by the IBM 4765 PCIe and IBM 4764 PCI-X Cryptographic Coprocessors. This allows the development of significant applications for CCA key management that will run without change on both systems.

ICSF's callable services enable installations to add cryptographic functions (such as MAC generation and verification) to current applications without redesign.

The combination of the hardware cryptographic features and ICSF provides high-performance cryptography, which removes bottlenecks on high-volume transaction applications and gives them needed protection. ICSF can support various combinations of cryptographic hardware.

An installation can use ICSF installation exits to change or extend the callable services.

# Protecting your investment

The use of an enterprise's computing resources is improved and protected by built-in product features.

ICSF also ensures that existing Program Cryptographic Facility (PCF) cryptographic applications, skills, and equipment can continue to be used effectively. This facilitates the earlier implementation of desired security applications while minimizing the disruption of existing applications.

- Existing PCF applications can run without change and without reassembly on ICSF in compatibility mode.
- A PCF conversion program has been provided to convert a PCF cryptographic key data set to an ICSF format.
- ICSF applications can run concurrently on the same processor with PCF applications.

# PCI-HSM compliance

PCI standards are developed by the PCI (Payment Card Industry) Security Standards Council to ensure security in the payment card industry. The PCI Security Standards Council defines their standards as "a set of security standards designed to ensure that all companies that accept, process, store, or transmit credit card information maintain a secure environment".

The goal of PCI-HSM is to improve security in payment card systems. It imposes requirements in key management, HSM API functions, device physical security, controls during manufacturing and delivery, device administration, and a number of other areas. It prohibits many things that were in common use for many years, but are no longer considered secure. The result of these requirements is that applications and procedures often must be updated because they used some of the things that are now prohibited.

Beginning with the Crypto Express6 adapter, a CCA coprocessor can be configured in PCI-HSM 2016 compliance mode. When in this mode, the "Payment Card Industry PIN Transaction Security Hardware Security Module Version 3.0, June 2016" standard is applied to applications identified as requiring compliance. The features and enhancements that are provided by the compliance mode implementation include:

- The ability to simultaneously support PCI-HSM compliant applications and non-compliant applications.
- Features to help you determine what parts of your current system need to be changed to be compliant.

- Mandatory dual control for sensitive operations.
- Separate logical key spaces to support both compliant and non-compliant workloads.
- Secure auditing of sensitive operations.
- Key usage restrictions for keys that are used in PCI-HSM compliant applications.
- Cryptographically protected information about firmware versions in the HSM, which can be viewed from a remote administration workstation.

These features and this environment are provided to support your needs when you are dealing with applications subject to PCI standards.

For information on:

- Upgrading applications to PCI-HSM compliance, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.
- Administering in a PCI-HSM compliant environment, see *z/OS Cryptographic Services ICSF Administrator's Guide*.
- Developing PCI-HSM compliant applications, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.
- PCI Security Standards Council and PCI standards, see PCI Security Standards (www.pcisecuritystandards.org).

# Auditing ICSF actions

There are events for which ICSF logs audit records by default. See "Event Recording" in *z/OS Cryptographic Services ICSF System Programmer's Guide* for more details about the SMF records that contain one or more user information sections. In addition to the audit records that are available by default, ICSF also provides options that can be used to gather different categories of events.

In addition, there may be settings within the SAF product which allows references to ICSF services and key labels to be audited. See "Controlling who can use cryptographic keys and services" in *z/OS Cryptographic Services ICSF Administrator's Guide* for more information.

## Cryptographic usage tracking

Use the **STATS(*value1[,...,value3]*)** option to enable usage tracking of cryptographic statistics at ICSF initialization. Exclude the **STATS(*value1[,...,value3]*)** option to disable cryptographic usage tracking.

**ENG**
  Enables usage tracking of cryptographic engines. Supports Crypto Express adapters, regional cryptographic servers, CPACF, and software.

**SRV**
  Enables usage tracking of cryptographic services. Supports ICSF callable services and UDXes only.

**ALG**
  Enables usage tracking of cryptographic algorithms. Supports cryptographic algorithms that are referenced in cryptographic operations. Limited support for key generation, key derivation, and key import.

The SMF records are in the form of SMF type 82 records. If an option is not specified, it is not tracked. For more information, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Use the STATSFILTERS(value) option to filter the criteria that is used to aggregate crypto usage statistics when STATS is enabled. Excluding this option means that ICSF uses all available criteria (that is, HOME job id, HOME job name, SECONDARY job name, HOME user id, task level user id, and ASID) to aggregate the crypto usage statistics.

**NOTKUSERID**
  Excludes the task level user id from the stats aggregation criteria. Enable this option in environments that have a high volume of operations that are running under task level user ids. This option reduces the number of SMF records written.

# Key lifecycle events

Use the AUDITKEYLIFECKDS, AUDITKEYLIFEPKDS, and AUDITKEYLIFETKDS options to audit the lifecycle of keys as they transition through the system. Keys can be audited from the time of their initial generation until their eventual disposal. A sample lifecycle of a key might be: key generated, key updated, key activated, key deactivated, key deleted.

AUDITKEYLIFECKDS controls the auditing of symmetric CCA keys, AUDITKEYLIFEPKDS controls the auditing of asymmetric CCA keys, and AUDITKEYLIFETKDS controls the auditing of PKCS #11 keys.

The audit records are in the form of SMF type 82 records. If an option is not specified, the default behavior is to audit updates to keys in the KDS. See *z/OS Cryptographic Services ICSF System Programmer's Guide* for more information.

# Key usage events

Use the AUDITKEYUSGCKDS, AUDITKEYUSGPKDS, and AUDITPKCS11USG options to audit the usage of keys.

AUDITKEYUSGCKDS controls the auditing of symmetric CCA keys, AUDITKEYUSGPKDS controls the auditing of asymmetric CCA keys, and AUDITPKCS11USG controls the auditing of PKCS #11 keys. The options accumulate repeated use of a key over an interval into a single audit record.

The audit records are in the form of SMF type 82 records. See *z/OS Cryptographic Services ICSF System Programmer's Guide* for more information.

# PKCS #11 FIPS-related events

The audit records generated as a result of the AUDITKEYLIFETKDS and AUDITPKCS11USG options include FIPS-related information where applicable. This information allows an installation to audit the FIPS parameters surrounding a request. See *z/OS Cryptographic Services ICSF System Programmer's Guide* for more information.

# Chapter 3. Application Programming Interfaces and key management

This topic describes the ICSF callable services and some of the concepts of cryptographic key management.

## Callable services

ICSF provides access to cryptographic functions through callable services. A callable service is a routine that receives control from a CALL statement in an application language. Each callable service performs one or more cryptographic functions or a utility function. Many of these callable services comply with IBM's Common Cryptographic Architecture (CCA), while others are extensions to the CCA.

The callable services available to your applications depend on your processor or server. For a list of the callable services available with each configuration, see Appendix B, "Summary of callable service support by hardware configuration," on page 63.

The ICSF Query Facility (CSFIQF) and ICSF Query Facility 2 (CSFIQF2) will return general information about ICSF. ICSF Query Facility (CSFIQF) also returns coprocessor information. The ICSF Query Algorithm (CSFIQA) returns the cryptographic and hash algorithms available.

The application programs can be written in high-level languages such as C, COBOL, FORTRAN, and PL/I, and in Assembler. ICSF callable services allow applications to perform these tasks:

- Enciphering and deciphering data by using the DES, TDES, AES algorithms. Many encryption modes are supported including cipher block chaining (CBC), Galois/Counter Mode (GCM), cipher feedback (CFB) and counter mode (CTR).
- Translating ciphertext from encryption under one key to encryption under another key by use of the Ciphertext translate callable service.

  This service securely deciphers the text that was enciphered under one key and then enciphers it under another key. The service supports many encryption modes and AES and DES algorithms.
- Generating DES cryptographic keys of all types for use by application programs.
- Generating AES cryptographic keys for use by application programs.
- Importing and exporting keys.
- Exchanging symmetric keys and their attributes with non-CCA systems using the TR-31 key block and TR-34 protocol.
- Generating PKA keys.

  Application programs can use the PKA key generate callable service to generate ECC and RSA private keys.
- Listing and deleting retained RSA private keys.

  Application programs can list and delete RSA private keys retained within the secure boundaries of a cryptographic feature coprocessors.
- Generating random numbers.

  Application programs can use a callable service to generate a random number for use in cryptography or for other general use. The callable service uses the cryptographic feature to generate a random number for use in encryption. The foundation for the random number generator is a time-variant input with a low probability of recycling.
- Generating and verifying PINs and translating PIN blocks.

**23**

An application program can use the callable services in generating and verifying PINs. In addition, use the Encrypted PIN translate callable service to reencrypt a PIN block from one PIN-encrypting key to another, or to reformat a PIN block.

- Generating and verifying DES MACs.

  An application can use MAC, MACVER, or DATA keys to generate and verify message authentication codes.

- Generating and verifying AES MACs.

  An application can use an AES DATA key to generate and verify message authentication codes.

- Generating and verifying HMAC MACs.

  An application can use an HMAC key to generate and verify message authentication codes.

- Generating MDCs, SHA-1, SHA-2 and other hashes.
- Generating and verifying Visa CVVs, MasterCard CVCs, Diner's Club CVVs, American Express CSCs.
- Developing EMV ICC applications.
- Enabling exploitation of clear key AES and DES encryption on CPACF supporting many modes of encryption.
- Writing Diffie-Hellman applications.
- Updating the CKDS and PKDS dynamically.

  ICSF provides callable services that application programs can use to create, read, write, and delete records in the CKDS and PKDS.

- Distributing DATA keys enciphered under an RSA key.
- Generating and verifying digital signatures.
- Composing and decomposing SET blocks.
- PKA-encrypting and PKA-decrypting any PKCS 1.2-formatted symmetric key data.

# Protecting and controlling DES keys

DES keys are protected by encryption under a DES master key (DES-MK). The DES master key always remains within the secure boundary of the cryptographic coprocessor on the server. There is only one DES master key and it is used only to encrypt and decrypt other DES keys. All coprocessors must have the same DES master key for the DES master key to become active.

**Note:** System without a cryptographic adapter, there is no encrypted key support.

The cryptographic hardware controls the use of DES keys by separating them into unique types. A unique key type can be used only for a specific purpose. For example, you cannot protect a key with a key that is intended to protect data. This hardware-enforced key separation provides better key protection than software key separation techniques.

**Note:** In ICSF, key separation applies to keys that are encrypted under the master key, as well as keys that are encrypted under transport key or key-encrypting keys. This enables the creator of a key to transmit the key to another system and to enforce its use at the other system.

## DES key forms

A key that is protected under the DES master key is in *operational form*, which means that ICSF can use it in cryptographic functions on the system.

When you store a key with a file or send it to another system, the key is enciphered under a transport key rather than the master key. When ICSF enciphers a key under a transport key, the key is not in operational form and cannot be used to perform cryptographic functions.

When a key is enciphered under a transport key, the sending system considers the key to be in the *exportable form*. The receiving system considers the key to be in the *importable form*. When a key is re-enciphered from under a transport key to under a system's master key, it is in operational form again.

## Control vectors

Control vectors contains key type, key management, and key usage attributes. For each type of DES key that the master key enciphers, there is a unique control vector. The control vector ensures that an operational key can only be used in cryptographic functions for which it is intended. For example, the control vector for an input PIN-encrypting key ensures that such a key can be used only in the PIN translation and PIN verification functions. "Types of DES keys" on page 25 describes the different DES key types.

For wrapping methods WRAP-ECB, WRAP-ENH, and WRAPENH2, the control vector is cryptographically bound to the key during the wrapping of the key. The length of the key is in the control vector in the key form bits (bits 40-42). For double-length keys, the left and right control vectors have different key form bits. For triple-length keys, the two control vectors are the same.

For the WRAPENH3 method, only one control vector is stored in the key token. The control vector is not used in the wrapping of the key. The key form bits are not used to determine the length of the key. The control vector is cryptographically bound to the key by the TDES-CMAC of the key token by a MAC key derived when the key is wrapped.

## Types of DES keys

ICSF groups DES cryptographic keys into these categories according to the functions they perform.

- DES Master key

  The DES master key is a double-length (128-bit) or triple-length (192-bit) key that is used only to encrypt other DES keys. The ICSF administrator installs and changes the DES master key using the ICSF panels. Alternatively, you can use the optional TKE workstation. The master key always remains within the secure boundary of the cryptographic feature. The DES master key is installed in the cryptographic coprocessors.

  The DES master key is used only to encipher and decipher operational keys. Cryptographic keys that are in exportable or importable form are not enciphered under the master key. They are enciphered under the appropriate transport key, which has itself been enciphered under the master key.

- Transport keys (or key-encrypting keys)

  Transport keys are also known as key-encrypting keys. They are double-length (128-bit) or triple-length (192-bit) keys that are used to protect keys when you distribute them from one system to another. For installations that do not support double-length 128-bit keys, ICSF supports the use of effective single-length keys. In an effective single-length key, the left half equals the right half.

  The DES transport keys are:

  – *EXPORTER or OKEYXLAT key-encrypting keys* protect keys of any type that are sent from your system to another. The exporter key at the originator is the same as the importer key of the receiver. An exporter key is paired with an importer key or a IKEYXLAT key.

  – *IMPORTER or IKEYXLAT key-encrypting keys* protect keys of any type that are sent from another system to yours. It also protects keys that you store externally in a file that you can import to your system at another time. The importer key at the receiver is the same as the exporter key at the originator. An importer key is paired with an exporter key or a OKEYXLAT key.

  **Note:** Transport keys replace the local, remote, and cross keys that PCF uses.

- Data-encrypting keys

  Data-encrypting keys are single-length (64-bit), double-length (128-bit), or triple-length (192-bit) keys that are used to encipher and decipher data. There are two classes of data-encrypting keys:

  **CIPHER:**
  The key types are CIPHER, ENCIPHER, and DECIPHER. These keys can only be used for enciphering and deciphering data. The key value is always encrypted.

**DATA:**
> The key type is DATA. This class of keys are used to encipher and decipher data. ICSF also provides support for the use of single-length and double-length keys in the callable services that generate and verify MACs. The key value can be encrypted or in the clear.

- Ciphertext translation keys

  These 128-bit keys are used for the Ciphertext Translate2 callable service as either the input or the output ciphertext translation key.

- MAC keys

  These can be single (64-bit), double-length (128-bit), or triple-length (192-bit) MAC and MACVER keys and double-length DATAM or DATAMV keys. These keys can be used to generate and verify MACs

- PIN keys

  The personal identification number (PIN) is a basis for verifying the identity of a customer across financial industry networks. PIN keys are double-length (128-bit) or triple-length (192-bit) keys. The callable services that generate, verify, and translate PINs use PIN keys.

  For installations that do not support double-length 128-bit keys ICSF provides effective single-length keys. In an effective single-length key, the left key half of the key equals the right key half.

- Key-generating keys

  Key-generating keys are used to derive unique-key-per transaction keys. These are double-length keys.

- Cryptographic variable encrypting keys

  These single-length keys are used to encrypt special control values in CCA DES key management. The Control Vector Translate and Cryptographic Variable Encipher callable services use cryptographic variable encrypting keys.

# Protecting and controlling AES keys

AES keys may be clear keys or secure keys protected by encryption under a AES master key. The AES master key always remains within the secure boundary of the cryptographic coprocessor on the server. There is only one AES master key and it is used only to encrypt and decrypt other AES keys and HMAC keys. All coprocessors must have the same AES master key for the AES master key to become active.

## AES key forms

A key that is protected under the AES-MK is in *operational form*, which means that ICSF can use it in cryptographic functions on the system.

When you store a key with a file or send it to another system, the key can be protected using an RSA key pair.

## Types of AES keys

ICSF groups AES cryptographic keys into these categories according to the functions they perform.

- AES Master key

  A 256-bit AES key that is used only to encrypt and decrypt AES or HMAC operational keys. The ICSF administrator installs and changes the AES master key using the ICSF panels or the optional TKE workstation. The AES master key always remains within the secure boundaries of the cryptographic coprocessors.

- Transport keys (or key-encrypting keys)

  Transport keys protect a key that is sent to another system, received from another system, or stored with data in a file. AES transport keys are variable-length keys up to 725 bytes in length.

  The AES transport keys are:

  – EXPORTER Key-encrypting Key

An EXPORTER key-encrypting key protects keys that are sent from your system to another system. The exporter key at the originator has the same clear value as the importer key at the receiver. An exporter key is paired with an importer key-encrypting key.

– IMPORTER Key-encrypting Key

An importer key-encrypting key protects keys that are sent from another system to your system. It also protects keys that you store externally in a file that you can import to your system later. The importer key at the receiver has the same clear value as the exporter key at the originator. An importer key is paired with an exporter key-encrypting key.

- Data-encrypting keys

Data-encrypting keys are used to encrypt and decrypt data. Data-encrypting keys can be 128-bits, 192-bits, or 256-bits in length. There are two classes of data-encrypting keys:

CIPHER:
This class of keys can be used for enciphering and deciphering data. These keys use the symmetric variable-length key token and have key usages attributes that can be used to restrict usage. The key value is always encrypted.

DATA:
This class of keys can be used for enciphering and deciphering data. These keys use the symmetric fixed-length key token. The key value can be either encrypted or in the clear.

- MAC keys

These keys can be used to generate and verify MACs. The CMAC algorithm is supported.

- Key-generating keys

Key-generating keys are used to derive unique-key-per transaction keys.

- PIN keys

The personal identification number (PIN) is a basis for verifying the identity of a customer across financial industry networks.

# Protecting and controlling HMAC keys

HMAC keys are protected by encryption under the AES master key. The AES master key always remains within the secure boundary of the cryptographic coprocessor on the server. There is only one AES master key and it is used only to encrypt and decrypt other AES and HMAC keys. All coprocessors must have the same AES master key for the AES master key to become active.

## HMAC key forms

A key that is protected under the AES-MK is in operational form, which means that ICSF can use it in cryptographic functions on the system.

When you store a key with a file or send it to another system, the key can be protected using an RSA key pair.

## HMAC keys

HMAC keys are variable length keys used to generate and verify MACs using the FIPS-198 Keyed-Hash Message Authentication Code (HMAC) algorithm.

# DES key token wrapping

ICSF wraps the key value in a DES key token using one of these methods.

WRAP-ECB
The original method of DES key wrapping has been used by ICSF since its initial release, and is the only key wrapping method that was available prior to FMID HCR7780. Using this original key wrapping

method, the key value in DES tokens are encrypted using triple DES encryption, and key parts are encrypted separately.

**WRAP-ENH**

The SHA-1 based enhanced method of symmetric key wrapping, introduced in FMID HCR7780, is ANSI X9.24 compliant. The wrapping key is derived using a NIST counter-mod key derivation function. SHA-1 HMAC is used in the KDF. Using the enhanced method, the key value for keys is bundled with other token data and encrypted using triple DES encryption and cipher block chaining mode. The enhanced method is available only on the z196 with a CEX3C or later coprocessor, and applies only to DES key tokens.

**WRAPENH2**

This method, which uses the enhanced wrapping method with SHA-256 HMAC KDF, was introduced in FMID HCR77C1. Using the enhanced method, the key value for keys is bundled with other token data and encrypted using triple DES encryption and cipher block chaining mode. The SHA-256 enhanced method applies only to triple-length key tokens. The SHA-256 enhanced method is available only on the z13 or z13s or later servers with the July 2019 licensed internal code (LIC) or the z14 or later servers with the December 2018 licensed internal code (LIC).

**WRAPENH3**

This method is based on the enhanced wrapping method using SHA-256 with the addition of an authentication code. The wrapping and MAC keys are derived using the NIST KDF with SHA-256 HMAC. A TDES-CMAC authentication code is generated over the complete key token. The authentication code is stored in the token where the right control vector was stored. There will always be three key parts encrypted and placed in the key token to obfuscate the key length. All keys with the exception of DATA keys with a zero control vector can be wrapped with this method. The method is available on z13 or z13s servers with the May 2021 licensed internal code, on z14 or z14 ZR1 servers with the May 2021 licensed internal code, or on z15 or z15 TO2 servers with the May 2021 licensed internal code.

The ICSF system programmer can specify the default wrapping method that ICSF will use for internal key tokens and external key tokens. The default wrapping method for internal key tokens and the default wrapping method for external key tokens are independent to each other and are specified separately. ICSF will use the specified method unless overridden by rule array keywords or by supplying a skeleton token with a different wrapping.

A CKDS conversion utility, CSFCNV2, enables you to convert all tokens in the CKDS to use the WRAP-ECB, WRAP-ENH, or WRAPENH3 wrapping methods.

If you are sharing a CKDS with other releases of ICSF and multiple IBM Z servers, you should select the enhanced wrapping method that is available for ICSF and the cryptographic adapters of your systems. Older releases and adapters cannot use key tokens wrapped by a method not supported.

## Protecting and controlling PKA keys

In a public key cryptographic system, it is a priority to maintain the security of the private key. It is vital that only the intended user or application have access to the private key.

On supported IBM servers, ICSF and the cryptographic hardware features ensure this by enciphering PKA private keys under a unique PKA object protection key. The PKA object protection key has itself been enciphered under a PKA master key. Each PKA private key also has a name that is cryptographically bound to the private key and cannot be altered. ICSF uses the private key name or the PKDS key label to control access to the private key. This combination of hardware-enforced coupling of cryptographic protection and access control, through the use of the Security Server (RACF), is unique to ICSF. It provides a significant level of security and integrity for PKA applications.

RSA keys may be stored in coprocessors providing additional security for PKA applications. You can generate RSA public and private key pairs within the secure hardware boundary of the coprocessor. In addition, you can retain the RSA private key within the feature where it is generated. The RSA private key is protected by the RSA master key.

## PKA master keys

The RSA master key is a 24-byte key used to encrypt the object protection key of RSA private keys. The ECC master key is a 32-byte key used to encrypt the object protection key of ECC and RSA private keys.

The ICSF administrator installs the PKA master keys on all cryptographic features by using either the ICSF pass phrase initialization panel, the master key entry panels, or the optional TKE workstation.

## RSA private and public keys

An RSA key pair includes a private and a public key. The RSA private key is used to generate digital signatures, and the RSA public key is used to verify digital signatures. The RSA public key is also used for key encryption of DES or AES DATA keys and the RSA private key for key recovery.

The RSA public key algorithm is based on the difficulty of the factorization problem. The factorization problem is to find all prime numbers of a given number, n. When n is sufficiently large and is the product of a few large prime numbers, this problem is believed to be difficult to solve. For RSA, n is typically at least 512 bits, and n is the product of two large prime numbers. For more information about the RSA public key algorithm, refer to the ISO 9796 standard.

### Generating RSA keys on a Cryptographic Coprocessor Feature

You can use the PKA key generate callable service to generate RSA public and private key pairs within the secure boundary of the cryptographic coprocessor. The modulus for the RSA keys may be up to 4096 bits depending on your system. The RSA private key may be retained and used within the secure boundary of the cryptographic coprocessor. The public key and the key name for the private key are stored in the ICSF public key data set (PKDS), but the value of a retained private key never appears in any form outside the cryptographic coprocessor.

## ECC private and public keys

An ECC key pair includes a private and public key. The ECC private key is used to generate digital signatures, and the ECC public key is used to verify digital signatures.

ICSF generates ECC key pairs using the Elliptic Curve Digital Signature Algorithm (ECDSA). This algorithm uses elliptic curve cryptography (an encryption system based on the properties of elliptic curves) to provide a variant of the Digital Signature Algorithm.

ECC keys are supported on the IBM z196 with a CEX3C or later feature. With a feature that is ECC capable, you can use the PKA key generate callable service to generate ECC keys.

## CRYSTALS-Dilithium private and public keys

A CRYSTALS-Dilithium key pair includes a private and public key. The CRYSTALS-Dilithium private key is used to generate digital signatures, and the CRYSTALS-Dilithium public key is used to verify digital signatures.

ICSF generates CRYSTALS-Dilithium key pairs using the CRYSTALS-Dilithium Digital Signature Algorithm. This algorithm uses lattice-based cryptography (an encryption system based on the hardness of finding short vectors in lattices) to provide a variant of the Digital Signature Algorithm.

CRYSTALS-Dilithium keys are supported on the IBM z15 with a CEX7S or later feature. With a feature that is CRYSTALS-Dilithium capable, you can use the PKA Key Generate callable service to generate CRYSTALS-Dilithium keys.

## Exchanging encrypted keys and PINs on a DES system

When a system sends a DATA key to another system, the sending system encrypts the DATA key under an *exporter* key-encrypting key. The receiving system re-encrypts the DATA key from encryption under an *importer* key-encrypting key to encryption under its master key. The importer and exporter key-encrypting keys at these systems complement each other and have the same clear value.
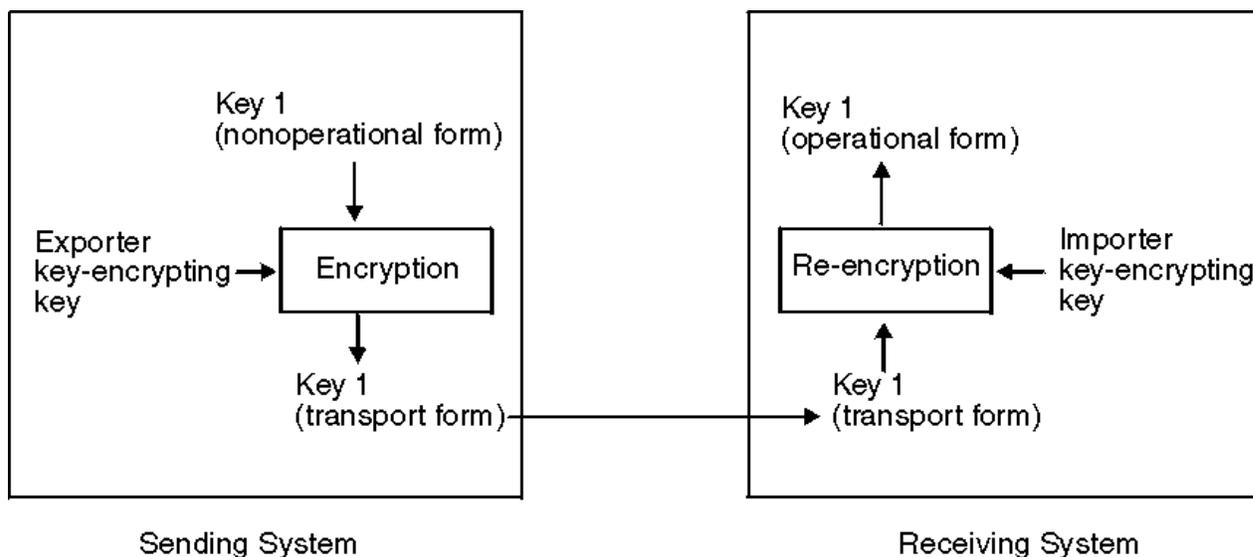
*Figure 8. Using transport keys to exchange keys*

In ICSF, you work with these complementary keys:

- Importer key-encrypting key and exporter key-encrypting key
- Importer key-encrypting key and OKEYXLAT key-encrypting key
- Exporter key-encrypting key and IKEYXLAT key-encrypting key
- Input PIN-encrypting key and output PIN-encrypting key
- PIN-generation key and PIN-verification key
- MAC-generation key and MAC-verification key

Your installation can use the key generator utility program (KGUP) or the callable services to generate and maintain complementary pairs of keys.

When KGUP generates a key, it also generates a KGUP control statement to create the complement of that key. You can send the control statement to the system with which you are exchanging keys or PINs.

## Exchanging RSA-encrypted data keys

In an RSA cryptographic system, the sending system and the receiving system do not need to share complementary importer and exporter key pairs to exchange DATA keys. The sender enciphers the DATA key by using the receiver's public key. The receiver deciphers the DATA key by using his or her own private key. Refer to "Exchanging DES or AES data-encrypting keys using an RSA key scheme" on page 12 for a more detailed explanation.

## Using multiple DES encipherment to protect keys and data

ICSF uses triple DES encipherment whenever they encipher a key under a key-encrypting key like the master key or a transport key. In addition to protecting and retrieving cryptographic keys, ICSF uses triple DES encipherment and decipherment to protect or retrieve 64-bit PIN blocks in the area of PIN applications. Triple DES encipherment is superior to single encipherment because it is much harder to break. The actual process to encipher a key depends on the type of key that is being enciphered and the type of key-encrypting key that is being used to encipher it.

Figure 9 on page 31 shows an example of triple DES encipherment. In this example, the left half of the enciphering key is used to encrypt the key in the first step. The result is then decrypted under the right half of the enciphering key. Finally, this result is encrypted under the left half of the enciphering key again.
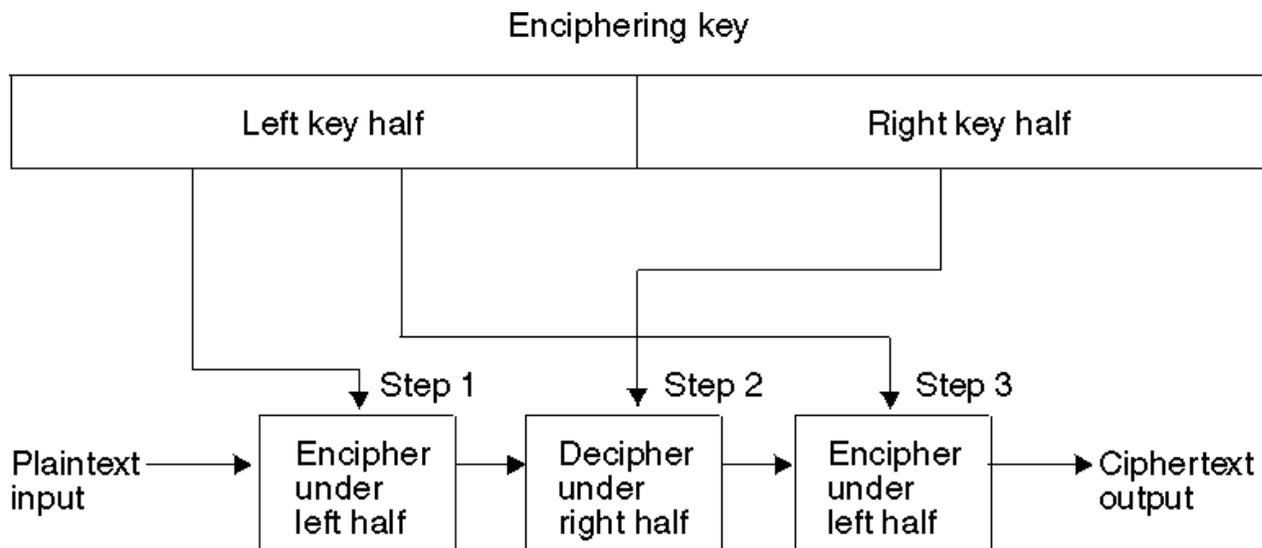
*Figure 9. An example of multiple encipherment*

ICSF uses triple DES data encipherment with either double-length or triple-length DATA keys to protect data. For this procedure the data is first enciphered using the first DATA key. The result is then deciphered using the second DATA key. When using a triple-length key, this second result is then enciphered using the third DATA key. When using a double-length key, the first DATA key is reused to encrypt the second result.

**Note:** Triple DES decipherment is the inverse of multiple encipherment (decipher-encipher-decipher).

# Running in special secure mode

Special secure mode is a special processing mode for the entry of clear keys. To perform these tasks, you must enable Special Secure Mode:

- Use the Secure Key Import, Secure Key Import2 or Multiple Secure Key Import callable services, which work with clear keys.
- Use the Clear PIN generate service which works with clear PINs.
- Use KGUP to enter clear keys into the CKDS.

Special secure mode is enabled by the SSM keyword in the installation options data set. Special secure mode can be enabled using the SAF profile CSF.SSM.ENABLE in the XFACILIT class. Additional hardware control for these callable services can be enforced with the optional TKE workstation.

# Cryptographic Key Data Set (CKDS)

ICSF stores AES, DES, and HMAC keys in a specialized data set called a cryptographic key data set (CKDS). ICSF maintains both a disk copy and an in-storage copy of the CKDS. This makes it possible to refresh the cryptographic keys without interrupting the application programs. ICSF provides a sample CKDS allocation job (members CSFCKDS, CSFCKD2, and CSFCKD3) in SYS1.SAMPLIB. An installation is not required to define a CKDS. However, when a CKDS is not defined, secure CCA symmetric key functions are unavailable and ICSF cannot be used to manage CCA symmetric key tokens. For more information on running in a sysplex environment, see *z/OS Cryptographic Services ICSF Administrator's Guide*.

ICSF updates the CKDS at these times:

- When you use KGUP to generate keys, to enter keys into the system or to load keys from a coprocessor, ICSF updates the disk copy, rather than the in-storage copy. ICSF does not require that you stop cryptographic functions before updating the CKDS, unlike PCF. After the update has been made, you can replace the in-storage copy of the CKDS with the disk copy using the ICSF panels.
- When you change the master key, ICSF enables you to reencipher the disk copy of the CKDS. ICSF then automatically refreshes the in-storage copy of the CKDS with the re-enciphered keys.

- When you convert a PCF CKDS to an ICSF CKDS, the PCF conversion program updates the disk copy of the ICSF CKDS.
- When an application uses the dynamic CKDS update callable services, both the disk copy and in-storage copy of the CKDS are dynamically updated.
- When a key is loaded using the Operational Key Load panel.
- When you convert a CKDS to use KDSR record format, the conversion program updates the disk copy of the ICSF CKDS.

ICSF allows these operations without interrupting cryptographic functions that are used by application programs.
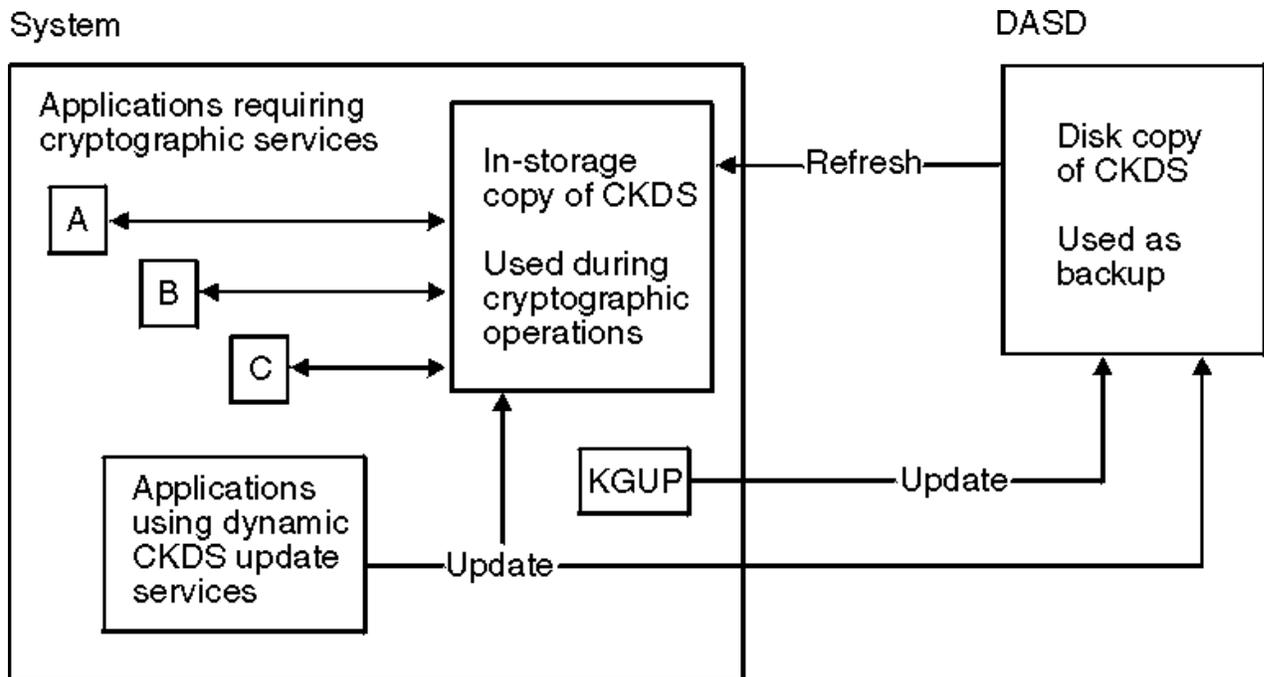


*Figure 10. How the cryptographic key data set is maintained and used*

Callable services use the in-storage copy of the CKDS. For example, in Figure 10 on page 32 applications A, B, and C might make many calls for services that require the CKDS. Having the CKDS in storage avoids time-consuming I/O to a data set that is stored on DASD.

KGUP updates the disk copy rather than the in-storage copy. The ICSF administrator can then use the ICSF panel dialog or a batch job to refresh the in-storage CKDS with the updated disk copy of the CKDS on every system sharing the updated CKDS. Cryptographic functions do not have to stop while KGUP updates the CKDS.

The dynamic CKDS update callable services permit an application to perform dynamic update of both the disk copy and the in-storage copy of the CKDS.

## Dynamic CKDS update callable services

The dynamic CKDS update callable services allow applications to directly manipulate both the in-storage copy and the DASD copy of the CKDS. These callable services have the identical syntax as the IBM 4765 PCIe and IBM 4764 verbs of the same name. Key management applications that use these common callable services, or verbs, can be run on either system without change. Cryptographic functions do not have to stop while the dynamic CKDS update callable services update the CKDS.

## Sysplex-wide consistency of CKDS

ICSF implements sysplex-wide consistent updates to the CKDS by using Cross-System Coupling Facility (XCF) signaling services and global (that is, sysplex-wide) ENQs. All members of the sysplex that are sharing the CKDS have their in-storage copy updated whenever the DASD copy of the CKDS is updated. All members that are sharing the CKDS have the same keys in their in-storage copy. The sysplex-wide coherency is enabled by the installation options data set SYSPLEXCKDS parameter. For more information, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

If the ICSF sysplex-wide coherency is not enabled, the in-storage copy of the CKDS is not the same for those members of the sysplex that are sharing the CKDS. The in-storage copy is updated only on the system that is initiating the CKDS update.

### Restrictions

The restrictions while using the sysplex-wide coherency support are:

- If multiple sysplexes or a sysplex and other non-sysplex system share a CKDS, there is no provision for automatic update of the in-storage copies of the CKDS on the systems that are not in the same sysplex as the system that is initiating the CKDS update.
- If KGUP is used to update the CKDS, the update is only made to the DASD copy of the CKDS. A refresh of the CKDS is required. A Coordinated CKDS Refresh refreshes all members of the sysplex that are sharing the CKDS. Otherwise, a local CKDS refresh is required on all systems that are sharing the updated CKDS.
- All sysplex members that are sharing a CKDS must change their master keys at the same time. The Coordinated Change Master Key utility can be used to change the symmetric master keys. All members that are sharing the CKDS are processed at the same time. For more information, see *z/OS Cryptographic Services ICSF Administrator's Guide*.
- The CKDS Entry Retrieval installation exit is not given control if SYSPLEXCKDS(YES,FAIL(xxx)) is coded in the ICSF Installation Options Data Set.

# Public Cryptographic Key Data Set (PKDS)

You can store RSA and ECC public and private keys, and trusted blocks in a specialized external VSAM data set that is called a public key data set (PKDS). ICSF maintains both a disk copy and an in-storage copy of the PKDS. This makes it possible to refresh the cryptographic keys without interrupting the application programs. ICSF provides a sample PKDS allocation job (member CSFPKDS) in SYS1.SAMPLIB. An installation is not required to define a PKDS. However, when a PKDS is not defined, secure CCA asymmetric key functions are unavailable and ICSF cannot be used to manage CCA asymmetric key tokens. For more information on running in a sysplex environment, see *z/OS Cryptographic Services ICSF Administrator's Guide*.

PKDS initialization support is available on the Master Key Management panel, CSFPUTIL utility, and PassPhrase Initialization. In order to enable PKA operations, the PKDS must be initialized.

## Dynamic PKDS update callable services

ICSF provides dynamic PKDS update callable services that permit an application to create, read, write, and delete PKDS records. You do not need to stop cryptographic functions while applications use these services to update the PKDS.

## Sysplex-wide consistency of PKDS

ICSF implements sysplex-wide consistent updates to the PKDS by using Cross-System Coupling Facility (XCF) signaling services and global (that is, sysplex-wide) ENQs. All members of the sysplex that are sharing the PKDS have their in-storage copy updated whenever the DASD copy of the PKDS is updated. All members that are sharing the PKDS have the same keys in their in-storage copy. The sysplex-wide coherency is enabled by the installation options data set SYSPLEXPKDS parameter. For more information, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

If the ICSF sysplex-wide coherency is not enabled, the in-storage copy of the PKDS is not the same for those members of the sysplex that are sharing the PKDS. The in-storage copy is updated only on the system that is initiating the PKDS update.

**Restrictions**

The restrictions while using the sysplex-wide coherency support are:

- If multiple sysplexes or a sysplex and other non-sysplex system share a PKDS, there is no provision for automatic update of the in-storage copies of the PKDS on the systems that are not in the same sysplex as the system that is initiating the PKDS update.
- All sysplex members that are sharing a PKDS must change their master keys at the same time. The Coordinated Change Master Key utility can be used to change the asymmetric master keys. All members that are sharing the PKDS are processed at the same time. For more information, see *z/OS Cryptographic Services ICSF Administrator's Guide*.

# Key Generator Utility Program and key generate callable service

With ICSF, you can use either the key generator utility program (KGUP) or the key generate callable service to generate DES or AES keys.

With KGUP, you can generate key-encrypting keys, PIN keys, data-encrypting keys, data-translation keys, and MAC keys. A master key variant enciphers each type of key that KGUP creates (except for CLRDES and CLRAES keys). After this program generates a key, it stores it in the CKDS where it can be saved and maintained.

The key generate callable service creates all types of DES or AES keys. It generates a single key or a pair of keys. Unlike KGUP, however, the key generate service does not store DES or AES keys in the CKDS but returns them to the application program that called it.

# Composing and decomposing SET blocks

ICSF provides callable services for developing SET applications that make use of the cryptographic hardware at the merchant and acquirer payment gateway. The SET Block Compose callable service performs DES encryption of data, OAEP-formatting through a series of SHA-1 hashing operations, and the RSA-encryption of the Optimal Asymmetric Encryption Padding (OAEP) block. The SET Block Decompose callable service decrypts both the RSA-encrypted and the DES-encrypted data.

# Exchanging Secure Sockets Layer session key seed

ICSF provides two callable services that make it possible to exchange the seed key that the SSL application needs to generate session keys. The PKA encrypt callable service encrypts a supplied clear key value under an RSA public key. Currently, this service supports the PKCS 1.2 and ZERO-PAD formats. The PKA decrypt callable service decrypts the supplied key value using the corresponding RSA private key and returns the seed key value to the application in the clear. Currently, this service supports only the PKCS 1.2 format. The SSL application can then use the clear key value to generate symmetric session keys.

# Enhanced key management for Crypto Assist instructions

ICSF can generate and build clear DES and AES tokens that can be used in callable services and stored in the cryptographic key data set (CKDS). Clear key tokens on the CKDS can be referenced by labelname by the Symmetric Key Encipher (CSNBSYE and CSNBSYE1) and the Symmetric Key Decipher (CSNBSYD and CSNBSYD1) services. With support for clear DES and AES keys in the CKDS, clear keys do not have to appear in application storage during use, allowing applications to exploit the performance of the CPACF with additional protection for the clear keys.

AES and DES clear keys can be generated using KGUP. A coprocessor is not required for clear key generation.

# Protected-key CPACF

Protected-key CPACF provides both high performance and high security by taking advantage of the high speed of CPACF while utilizing encrypted keys. It does this by using CPACF wrapping keys to protect the key during CPACF processing instead of passing a clear key. These wrapping keys (one for Advanced Encryption Standard (AES) keys and one for Data Encryption Standard (DES) keys) are analogous to the coprocessor master keys and are visible only to licensed internal code (LIC) and never to operating system storage.

Five callable services support protected-key CPACF:

- CKDS Key Record Read2 (CSNBKRR2 and CSNEKRR2)
- Field Level Encipher (CSNBFLE and CSNEFLE)
- Field Level Decipher (CSNBFLD and CSNEFLD)
- Symmetric Key Encipher (CSNBSYE, CSNBSYE1, CSNESYE, CSNESYE1)
- Symmetric Key Decipher (CSNBSYD, CSNBSYD1, CSNESYD, CSNESYD1)

Field Level Encipher, Field Level Decipher, Symmetric Key Encipher, and Symmetric Key Decipher accept labels for the *key_identifier* parameter when the KEYIDENT keyword is provided in the *rule_array*. Before protected-key CPACF, this label was restricted to refer to a clear DATA key in the CKDS. With protected-key CPACF enabled, the label may now refer to an encrypted DATA key as well. Field Level Encipher and Field Level Decipher additionally support an encrypted DATA key token that does not reside in the CKDS for the *key_identifier* parameter.

CKDS Key Record Read2 with the PROTKEY rule returns the protected-key CPACF form of the CCA token to a caller with sufficient authority (either system key or supervisor state).

ICSF processes a secure key usable by a coprocessor (a CCA encrypted key token) into a secure key usable by CPACF (a CPACF-wrapped key). Each CPACF wrapped key is kept on hand after the first use so it can be used again for a subsequent encryption or decryption request.

To transform a CCA-encrypted key token into a CPACF-wrapped key, ICSF does the following:

1. Determines if the key has already been wrapped for use with CPACF. ICSF maintains a cache of CPACF-wrapped DATA keys by label. When a label is specified on a call to the Symmetric Key Encipher or Symmetric Key Decipher service or when a label or token is specified on a call to the Field Level Encipher or Field Level Decipher service, ICSF retrieves the key from the in-storage copy of the CKDS or protected key token cache. If it is an encrypted DATA key, ICSF looks for a cached copy and uses it if one is present.

2. Determines if this key is a candidate for wrapping. If the key has not been wrapped for CPACF and cached, ICSF inspects a field in the covering CSFKEYS profile to check for permission. A CSFKEYS profile can contain an ICSF segment, which specifies rules for key use. The SYMCPACFWRAP field of the ICSF segment indicates whether ICSF can rewrap the encrypted key using the CPACF wrapping key. If there is no covering profile, or ICSF(SYMCPACFWRAP(NO)) is set, ICSF does not allow the operation. Additionally, for CKDS Key Record Read2 with the PROTKEY rule, the SYMCPACFRET field of the ICSF segment is checked to determine whether ICSF can return the protected-key CPACF form.

3. Requests the wrapping operation. ICSF builds a request to a Crypto Express3 Coprocessor (CEX3C) or later coprocessor. In the coprocessor, the encrypted DATA key is recovered from under the card master key. The clear form is presented back to the LIC layer, which wraps the clear key value under the corresponding CPACF wrapping key (either AES or DES) before returning the key to operating system storage. At no point during this operation is the clear key value visible in operating system storage.

4. Caches the returned CPACF-wrapped key for future use.

Figure 11 on page 36 shows how ICSF transforms a CCA-encrypted key token into a CPACF-wrapped key.
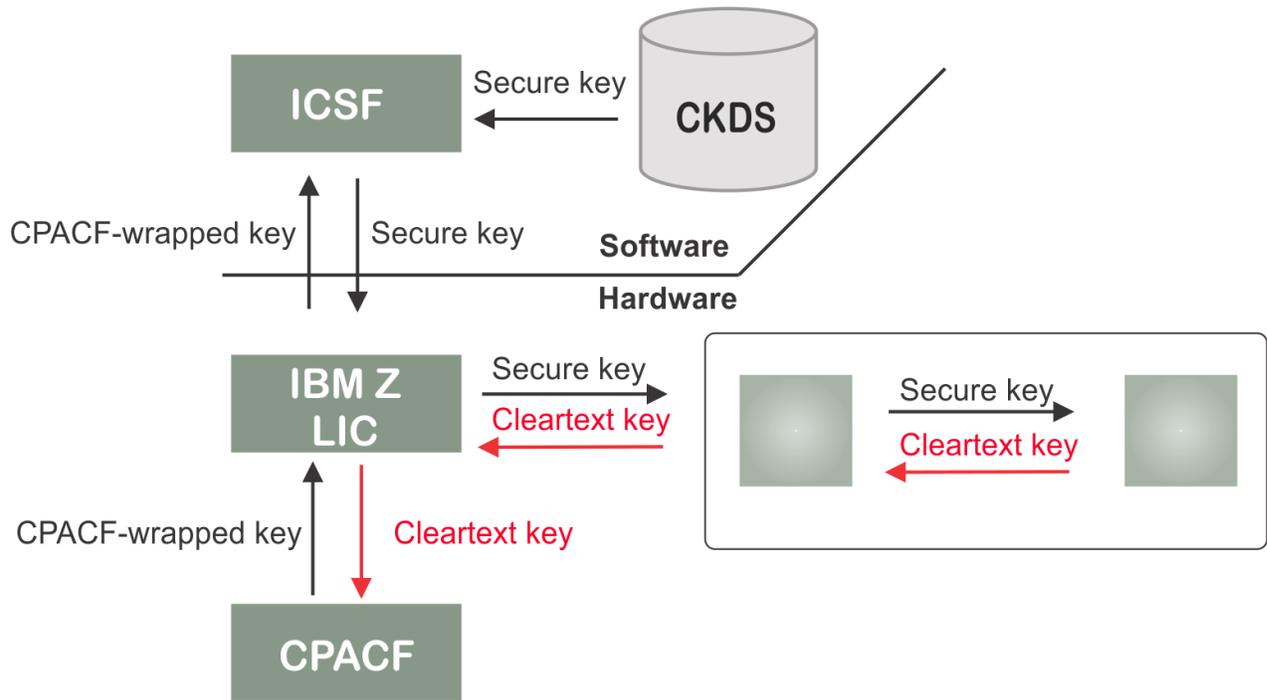
*Figure 11. Transforming a CCA-encrypted key token into a CPACF-wrapped key*

For more information about the Field Level Encipher and Field Level Decipher callable services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

For more information on the SYMCPACFWRAP and SYMCPACFRET fields of the ICSF segment of CSFKEYS profiles, see *z/OS Cryptographic Services ICSF Administrator's Guide*.

# PKCS #11

PKCS #11, also known as Cryptoki, is the cryptographic token interface standard. It specifies an application programming interface (API) to devices, referred to as tokens, that hold cryptographic information and perform cryptographic functions. The PKCS #11 API is an industry-accepted standard commonly used by cryptographic applications. ICSF supports PKCS #11, providing an alternative to IBM's Common Cryptographic Architecture (CCA) and broadening the scope of cryptographic applications that can make use of zSeries cryptography. PKCS #11 applications developed for other platforms can be recompiled and run on z/OS.

The PKCS #11 standard can be found at PKCS#11: Cryptographic Token Interface Standard (www.cryptsoft.com/pkcs11doc). This topic describes how ICSF supports that standard. The support includes:

- A token data set (TKDS) that serves as a repository for cryptographic keys and certificates used by PKCS #11 applications
- A C application programming interface (API) that supports a subset of the V2.20 level of the PKCS #11 specification
- Token management callable services. The C API uses these callable services.

## Tokens

On most single-user systems a token is a smart card or other plug-installed cryptographic device, accessed through a card reader or slot. The PKCS #11 specification assigns numbers to slots, known as slot IDs. An application identifies the token that it wants to access by specifying the appropriate slot ID. On systems that have multiple slots, it is the application's responsibility to determine which slot to access.

z/OS must support multiple users, each potentially needing a unique keystore. In this multiuser environment, the system does not give users direct access to the cryptographic cards installed as if they were personal smart cards. Instead, z/OS PKCS #11 tokens are virtual, conceptually similar to RACF (SAF) key rings. An application can have one or more z/OS PKCS #11 tokens, depending on its needs.

Typically, PKCS #11 tokens are created in a factory and initialized either before they are installed or upon their first use. In contrast, z/OS PKCS #11 tokens can be created using system software such as RACF, the gskkyman utility, or by applications using the C API. Each token has a unique token name, or label, that is specified by the end user or application at the time that the token is created.

In addition to any tokens your installation may create, ICSF creates a token that is available to all applications. This "omnipresent" token is created by ICSF in order to enable PKCS #11 services when no other token has been created. In this situation, key types and cryptographic mechanisms are available in software. The token label for the omnipresent token is SYSTOK-SESSION-ONLY.

Because PKCS #11 tokens are typically physical hardware devices, the PKCS #11 specification provides no mechanism to delete tokens. However, because z/OS PKCS #11 tokens are virtual, z/OS must provide a way to delete them. To delete a z/OS PKCS #11 token, call C_InitToken with a special label value, $$DELETE-TOKEN$$ (assuming code page IBM1047).

## Token Data Set (TKDS)

ICSF stores the PKCS #11 tokens and token objects in a specialized data set called the token data set (TKDS). ICSF maintains both a disk copy and an in-storage copy of the TKDS. This makes it possible to refresh the PKCS #11 tokens and objects without interrupting the application programs. ICSF provides a sample TKDS allocation job (members CSFTKDS, CSFTKD2) in SYS1.SAMPLIB.

A TKDS is no longer required in order to run PKCS #11 applications. If ICSF is started without a TKDS, however, only the omnipresent token will be available.

Callable services use the in-storage copy of the TKDS. Having the TKDS in storage avoids time-consuming I/O to a data set that is stored on DASD. The dynamic TKDS update callable services permit an application to perform dynamic update of both the disk copy and the in-storage copy of the TKDS.

ICSF supports sysplex-wide consistent updates to the TKDS through the use of Cross-System Coupling Facility (XCF) signalling services and global (that is, sysplex-wide) ENQs. This support maintains the consistency of the in-storage TKDS in a sysplex environment. If a TKDS record is modified by create, update, or delete operations, the DASD version of the TKDS is updated and the ICSF in-storage copy is updated to reflect the new contents of the record for all systems in the ICSF sysplex group.

## PKCS #11 and FIPS 140-2

The National Institute of Standards and Technology (NIST), the US federal technology agency that works with industry to develop and apply technology, has published the Federal Information Processing Standard Security Requirements for Cryptographic Modules standard (FIPS 140-2), that can be required by organizations who specify that cryptographic-based security systems are to be used to provide protection for sensitive or valuable data.

The z/OS PKCS #11 services are designed to meet FIPS 140-2 Level 1 criteria, and can be configured to operate in compliance with FIPS 140-2 specifications. Applications that need to comply with the FIPS 140-2 standard can therefore use the z/OS PKCS #11 services in a way that allows only the cryptographic algorithms (including key sizes) approved by the standard and restricts access to the algorithms that are not approved. There are two modes of FIPS-compliant operation:

- The services can be configured so that all z/OS PKCS #11 applications are forced to comply with the FIPS 140-2 standard.
- For installations where only certain z/OS PKCS #11 applications need to comply with the FIPS 140-2 standard, the services can be configured so that only the necessary applications are restricted from using the non-approved algorithms, while other applications are not.

For more information on PKCS #11 and FIPS 140-2 standards, refer to *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*.

# Quantum-safe cryptography

Quantum-safe cryptography includes a suite of algorithms that are resistant to attacks by both classical and quantum computers. Traditional public-key cryptography relies upon mathematical problems that are difficult to solve on classical computers. However, popular cryptographic schemes, such as RSA and ECC, can be easily broken by a sufficiently large quantum computer.

ICSF supports the following quantum-safe algorithms (QSA):

- CRYSTALS-Dilithium Digital Signature Algorithm.

## CRYSTALS-Dilithium Digital Signature Algorithm

CRYSTALS-Dilithium is a lattice-based digital signature scheme whose security is based on the hardness of finding short vectors in lattices. The CRYSTALS-Dilithium Digital Signature Algorithm is a member of the CRYSTALS (Cryptographic Suite for Algebraic Lattices) suite of algorithms. The strength of a CRYSTALS-Dilithium key is represented by the size of its matrix of polynomials. For example, CRYSTALS-Dilithium (6,5) has a matrix size of 6x5. The larger the matrix size, the stronger the key. CRYSTALS-Dilithium keys can only be used for Digital Signature Generation and Verification.

ICSF supports the CRYSTALS-Dilithium Signature Algorithm on both the PKCS#11 and CCA architectures. PKCS#11 CRYSTALS-Dilithium key operations can be performed in hardware or software. CRYSTALS-Dilithium key operations are supported on the IBM z15 with a CEX7S or later feature. CCA CRYSTALS-Dilithium keys cannot be currently be stored in a KDS, but PKCS#11 CRYSTALS-Dilithium keys can be stored in the TKDS. There is no C-API for CRYSTALS-Dilithium keys. The abbreviation, LI2, is used to refer to CRYSTALS-Dilithium in character restricted fields.

PKCS#11 callable services that support CRYSTALS-Dilithium key operations are:

- PKCS #11 Generate Key Pair (CSFPGKP and CSFPGKP6).
- PKCS #11 One-Way Hash, Sign, or Verify (CSFPOWH and CSFPOWH6).
- PKCS #11 Private Key Sign (CSFPPKS and CSFPPKS6).
- PKCS #11 Public Key Verify (CSFPPKV and CSFPPKV6).
- PKCS #11 Token Record Create (CSFPTRC and CSFPTRC6).

CCA callable services that support CRYSTALS-Dilithium key operations are:

- Digital Signature Generate (CSNDDSG and CSNFDSG).
- Digital Signature Verify (CSNDDSV and CSNFDSV).
- PKA Key Generate (CSNDPKG and CSNFPKG).
- PKA Key Import (CSNDPKI and CSNFPKI).
- PKA Key Token Build (CSNDPKB and CSNFPKB).
- PKA Key Token Change (CSNDKTC and CSNFKTC).
- PKA Public Key Extract (CSNDPKX and CSNFPKX).

# Chapter 4. Using ICSF with other cryptographic products

This topic describes how ICSF works with other cryptographic products.

## Using IBM's Common Cryptographic Architecture

ICSF provides callable services that comply with IBM's Common Cryptographic Architecture (CCA). This allows application programs written in high-level languages such as C, COBOL, FORTRAN, and PL/I, as well as in Assembler, to be used under more than one cryptographic product.

Another family of products that provide these services is the IBM 4765 PCIe, IBM 4767 PCIe, and IBM 4764 PCI-X cryptographic coprocessors.

## Coexisting with other IBM cryptographic products

ICSF can coexist simultaneously with other IBM cryptographic products within the same operating system image. This protects your installation's investment in programming skills and user applications, and provides a framework for migrating to ICSF.

### Running PCF applications under ICSF

If your installation uses PCF, you can run PCF applications on ICSF. Your applications can benefit from the enhanced performance and availability of ICSF. Running PCF applications on ICSF allows you to test ICSF. ICSF also helps you migrate PCF applications. As soon as you can, you should convert these applications to use ICSF callable services rather than the PCF macros. This will permit you to change the master key without interrupting the converted applications.

ICSF continues to support the PCF macros (GENKEY, RETKEY, EMK, and CIPHER). If an application uses these PCF macros, you can run the application on ICSF. The CIPHER macro will use the DES algorithm. If exits exist for either the GENKEY or the RETKEY macro, you should evaluate their applicability to ICSF. If your applications still need these exits, you must rewrite them for ICSF.

You can run PCF applications on systems with ICSF installed. How they run depends on the mode in which ICSF is running. You can run ICSF in any of these modes:

- In **compatibility mode**, you can run PCF applications on ICSF without reassembling them, because ICSF supports the PCF macros.

  You cannot start PCF at the same time as ICSF on the same operating system.

- In **coexistence mode**, you can run a PCF application on PCF, or you can reassemble it to run on ICSF. ICSF provides coexistence macros for this purpose.

  You can start PCF at the same time as ICSF on the same operating system.

- In **noncompatibility mode**, you can run PCF applications only on PCF, and you can run ICSF applications only on ICSF. You cannot run PCF applications on ICSF because ICSF does not support the PCF macros in this mode.

  You can start PCF at the same time as ICSF on the same operating system.

An application that is running under PCF may use a key in a CKDS managed by PCF. Before you run such an application on ICSF, you should convert the key to an ICSF format. ICSF provides a program to do this conversion.

You should use noncompatibility mode unless you are migrating from PCF to ICSF.

# Managing keys with the Distributed Key Management System (DKMS)

The Distributed Key Management System (DKMS) provides online key management to ICSF as well as to IBM cryptographic products on other platforms. DKMS offers centralized key management for symmetric and asymmetric keys and for certificates. DKMS automates the key management process, and exchanges and replaces keys and certificates on demand. Further, to assure continuous operation DKMS maintains backup copies of all critical keys.

With the introduction of Payment Card Industry Data Security Standard (PCI DSS) and other guidelines and requirements key management is much more than a suitable toolbox. DKMS helps enforcing the organization's policies and procedures by offering dual control, split knowledge, audit trail, and key separation between applications and production environments.

The DKMS system is comprised of a workstation that constitutes the user interface and a server component (the DKMS agent) that interacts with ICSF and the backup key repository. The architecture allows for multiple agents, supporting simultaneous management of keys on several servers. The applications get cryptographic support by using ICSF callable services or by utilizing high level DKMS application programming interfaces.

In addition to essential management of symmetric and asymmetric keys, DKMS offers a number of business-focused features to meet specific needs. These features include:

- **Certificate Management**

  DKMS manages certificates stored in RACF Key Rings that are accessible online and certificates used by SSL servers or other connection implementations that must be addressed offline.

  DKMS supports all aspects of RACF Key Ring administration. Key Rings can be created or deleted. Further, new keys can be generated and these keys, together with their certificates, can be attached to one or more key rings. All functions are performed online from the DKMS workstation.

  Many web services and other communication connections rely on a RSA based certificate scheme to assure authenticity and privacy. This scheme requires that RSA keys and certificates are renewed at regular intervals. The DKMS SSL certificate management feature centralizes and unifies most of the tasks traditionally performed manually for components utilizing SSL or other certificate based schemes. Further, functions are offered that ease administration of certificates for a large population of SSL servers. The DKMS SSL certificate management supports numerous SSL server implementations.

- **EMV support**

  DKMS supports all parties of EMV business: Brand Certificate Authorities (CAs), Integrated Circuit Card (ICC) card issuers, and ICC transaction acquirers.

  - Brand CAs totally support all the security and procedural requirements needed to implement CAs to Visa and MasterCard specifications.

  - ICC card issuers supports generation of the key material to be stored in the EMV ICC card for both the SDA and the DDA/CDA schemes. For the DDA/CDA schemes, DKMS implements a key pre-generation mechanism that utilizes low-load periods of ICSF's cryptographic coprocessors for generation of the huge amount of RSA keys required.

  - ICC transaction acquirers has transaction authorization support for verification of application cryptograms, generation of response cryptograms and secure scripts.

- **Remote Key Loading for ATMs**

  Contemporary ATMs support keys to be exchanged with back-end systems using an RSA key exchange scheme defined in ANS X9.24 part 2. DKMS provides all functions to support this scheme. This includes exchange of keys and certificates with the ATM vendors and APIs that supplies the ATM keys in a format suitable for the ATMs. The DKMS RKL feature supports the major ATM vendors.

For further information, contact the Crypto Competence Center, Copenhagen (www.ibm.com/security/key-management).

# Encrypting and decrypting information from other products

ICSF can exchange encrypted information with other cryptographic products. The only limitation is the form of DES encryption used. Some examples:

- MACs: ICSF supports both the ANSI standard X9.9, option 1, and the X9.19 optional double-MAC procedure for generating message authentication codes (MACs). Therefore, if a MAC has been generated with another product that uses either of these standards, ICSF can verify that MAC. ICSF provides support for the use of data-encrypting keys in both the MAC generating and verifying services. This support allows these services to interface more smoothly with non-CCA key distribution systems, including those that follow the ANSI X9.17 protocol.

- PINs: ICSF supports a wide variety of PIN block formats, as is shown in "Using Personal Identification Numbers (PINs) for personal authentication" on page 14.

- Data: ICSF can exchange encrypted data with other products that use the cipher block chaining (CBC) form of the DES and AES algorithms.

- Keys: ICSF can exchange encrypted keys with other products that conform to the IBM's Common Cryptographic Architecture. If you need to exchange keys with systems that do not recognize transport key variants, ICSF enables you to encrypt selected keys under the transport key rather than under the transport key variant. You can use either an application program or KGUP to do this. ICSF can exchange RSA-encrypted data-encrypting keys with systems that format the key according to the PKCS 1.2 Standard. Remote Key Loading can be used to communicate with non-CCA compliant devices. ICSF supports importing and exporting DES keys in the ANSI TR-31 key block.

- Digital Signatures: ICSF can exchange digital signatures with systems that support any of these standards:
  - RSA signatures with MDC, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 or MD5 hashes and ISO-9796 formatting
  - RSA signatures with MD5 hashes and PKCS 1.0 or PKCS 1.1 formatting
  - ANSI X9.62 (ECDSA)

# Encryption facility

## What is encryption facility?

The need for creating secure archived copies of business data is a critical security concern. Encrypting data that can be recovered at any time offers a high degree of privacy protection from unwanted access. Encryption Facility provides this protection by offering encryption of data for exchange between different systems and platforms and for archiving purposes. It makes use of hardware compression and encryption and relies on a centralized key management based on the z/OS Integrated Cryptographic Service Facility (ICSF) that is highly secure and easy to use.

Encryption Facility makes use of ICSF to perform encryption and decryption and to manage cryptographic keys. To encrypt data files Encryption Facility uses these kinds of cryptographic keys:

- TDES triple-length keys
- 128-bit AES keys

**Note:** The IBM Z format can use secure symmetric keys.

For information about cryptographic keys, see *z/OS Cryptographic Services ICSF Administrator's Guide* and *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

## Features available with encryption facility

Version 1 Release 2.0 of IBM Encryption Facility for z/OS provides these optional features:

| Table 1. Features for Encryption Facility | |
|---|---|
| **Feature** | **Description** |
| **IBM Encryption Facility for z/OS Encryption Services, called Encryption Services** | Support for openPGP and complementary encryption and decryption batch programs that run on z/OS and allow you to encrypt and decrypt data. The algorithms that can be used are:<br><br>• AES 128, 192 and 256<br>• 3DES<br>• blowfish |
| **IBM Encryption Facility for z/OS DFSMSdss Encryption, called DFSMSdss Encryption** | Services that run on z/OS DFSMSdss and allow you to use DFSMSdss commands to encrypt and decrypt data. With this feature, you can also use DFSMShsm. |

For more information about how Encryption Services works, see *IBM Encryption Facility for z/OS: Using Encryption Facility for OpenPGP*.

# Virtual Telecommunications Access Method (VTAM) session-level encryption

ICSF supports VTAM session-level encryption, which provides protection for messages within SNA sessions—that is, between pairs of logical units.

When VTAM session- level encryption is in effect, only the originating logical unit can encipher the data, and only the destination logical unit can decipher the data. Thus, the data never appears in the clear while passing through the network.

ICSF places no restrictions on the addressing mode of calling programs. In particular, when VTAM session-level encryption is used with ICSF, VTAM can use storage above 16 megabytes.

# Access Method Services Cryptographic Option

ICSF supports the Access Method Services Cryptographic Option. The option enables the user of the Access Method Services REPRO command to encipher data by using the Data Encryption Algorithm. The Access Method Services user can use REPRO to encipher data, write it to a data set, and then store the enciphered data set offline. When the user needs the enciphered data set, he or she can retrieve it and use REPRO to decipher it. The user can decipher the data either on the host processor where it was enciphered or on another host processor that contains the Access Method Services Cryptographic Option and the cryptographic key needed.

With the exception of catalogs, all data set organizations that are supported for input by REPRO are eligible as input for enciphering. Similarly, and with the same exception, all data set organizations supported for output by REPRO are eligible as output for deciphering. The resulting enciphered data sets are always sequentially organized (SAM or VSAM entry-sequenced data sets).

Cryptographic keys can either be created by ICSF or be supplied by the Access Method Services user.

# Chapter 5. Planning for the Integrated Cryptographic Service Facility

This topic contains guidelines and suggestions to help you plan the installation and operation of ICSF.

## System requirements

ICSF is an element of z/OS, but does not always follow the z/OS release schedule. It sometimes provides independent ICSF releases as web deliverables. These are identified by their FMID.

ICSF FMID HCR77D1 runs on zSeries servers: z9 EC, z9 BC, z10 EC, z10 BC, z196, z114, zEC12, zBC12, z13, z13s, z14, z14 ZR1, and z15.

### z/OS ICSF FMIDs

These tables explain the relationships of z/OS releases, ICSF FMIDs and servers.

Table 2. z/OS ICSF FMIDs

| z/OS | ICSF FMID | Web deliverable name |
|---|---|---|
| **V2R2** | HCR77B0 | Enhanced Cryptographic Support for z/OS V1R13 - z/OS V2R1. |
| | HCR77B1 | Cryptographic Support for z/OS V1R13 - z/OS V2R2. |
| | HCR77C0 | Cryptographic Support for z/OS V2R1 - z/OS V2R2. |
| | HCR77C1 | Cryptographic Support for z/OS V2R1 - z/OS V2R3. |
| | HCR77D0 | Cryptographic Support for z/OS V2R2 - z/OS V2R3 |
| | HCR77D1 | Cryptographic Support for z/OS V2R2 - z/OS V2R4 |
| **V2R3** | HCR77C0 | Cryptographic Support for z/OS V2R1 - z/OS V2R2. |
| | HCR77C1 | Cryptographic Support for z/OS V2R1 - z/OS V2R3. |
| | HCR77D0 | Cryptographic Support for z/OS V2R2 - z/OS V2R3 |
| | HCR77D1 | Cryptographic Support for z/OS V2R2 - z/OS V2R4 |
| **V2R4** | HCR77D0 | Cryptographic Support for z/OS V2R2 - z/OS V2R3 |
| | HCR77D1 | Cryptographic Support for z/OS V2R2 - z/OS V2R4 |

Refer to this chart to determine what release is associated with each ICSF FMID and what server it will run on.

Table 3. FMID and Hardware

| ICSF FMID | Applicable z/OS Releases | Servers where FMID will run |
|---|---|---|
| **HCR77B0 (Base of z/OS 2.2)** | 1.13, 2.1, and 2.2 | z890, z990, z9 EC, z9 BC, z10 EC, z10 BC, z114, z196, zBC12, zEC12, z13, z13s, z14, z14 ZR1, and z15. |
| **HCR77B1** | 1.13, 2.1, and 2.2 | z890, z990, z9 EC, z9 BC, z10 EC, z10 BC, z114, z196, zBC12, zEC12, z13, z13s, z14, z14 ZR1, and z15. |

| Table 3. FMID and Hardware (continued) | | |
|---|---|---|
| **ICSF FMID** | **Applicable z/OS Releases** | **Servers where FMID will run** |
| **HCR77C0 (Base of z/OS 2.3)** | 2.1, 2.2, and 2.3 | z9 EC, z9 BC, z10 EC, z10 BC, z114, z196, zBC12, zEC12, z13, z13s, z14, z14 ZR1, and z15. |
| **HCR77C1** | 2.1, 2.2, and 2.3 | z9 EC, z9 BC, z10 EC, z10 BC, z114, z196, zBC12, zEC12, z13, z13s, z14, z14 ZR1, and z15. |
| **HCR77D0** | 2.2 and 2.3 | z9 EC, z9 BC, z10 EC, z10 BC, z114, z196, zBC12, zEC12, z13, z13s, z14, z14 ZR1, and z15. |
| **HCR77D1** | 2.2, 2.3, and 2.4 | z9 EC, z9 BC, z10 EC, z10 BC, z114, z196, zBC12, zEC12, z13, z13s, z14, z14 ZR1, and z15. |

# Migration information

The migration topic is covered in detail in *z/OS Cryptographic Services ICSF System Programmer's Guide*.

# Cryptographic hardware features

This topic describes the cryptographic hardware features available. Information on adding and removing cryptographic coprocessors can be found in *z/OS Cryptographic Services ICSF Administrator's Guide*.

### Crypto Express7 adapter (CEX7C, CEX7P, or CEX7A)

The Crypto Express7 adapter is an asynchronous cryptographic coprocessor or accelerator. The adapter contains one cryptographic engine that can be configured as a coprocessor (CEX7C for CCA and CEX7P for PKCS #11) or as an accelerator (CEX7A). One feature may include one or two adapters, depending on the feature code. It is available on z15.

### Crypto Express6 adapter (CEX6C, CEX6P, or CEX6A)

The Crypto Express6 adapter is an asynchronous cryptographic coprocessor or accelerator. The adapter contains one cryptographic engine that can be configured as a coprocessor (CEX6C for CCA and CEX6P for PKCS #11) or as an accelerator (CEX6A). It is available on IBM z14, IBM z14 ZR1, and IBM z15.

### Crypto Express5 adapter (CEX5C, CEX5P, or CEX5A)

The Crypto Express5 adapter is an asynchronous cryptographic coprocessor or accelerator. The adapter contains one cryptographic engine that can be configured as a coprocessor (CEX5C for CCA and CEX5P for PKCS #11) or as an accelerator (CEX5A). It is available on IBM z13, IBM z13s, IBM z14, IBM z14 ZR1, and IBM z15.

### Crypto Express4 adapter (CEX4C, CEX4P, or CEX4A)

The Crypto Express4 adapter is an asynchronous cryptographic coprocessor or accelerator. The adapter may be configured as a CCA coprocessor (CEX4C), an Enterprise PKCS #11 coprocessor (CEX4P), or as an accelerator (CEX4A). It is available on IBM zEnterprise EC12 and IBM zEnterprise BC12.

### Crypto Express3 adapter (CEX3C or CEX3A)

The Crypto Express3 adapter is an asynchronous cryptographic coprocessor or accelerator. The adapter contains two cryptographic engines that can be independently configured as a coprocessor (CEX3C) or as

an accelerator (CEX3A). It is available on the IBM System z10 Enterprise Class, IBM System z10 Business Class, IBM zEnterprise 196, IBM zEnterprise 114, IBM zEnterprise EC12, and the IBM zEnterprise BC12.

### Crypto Express2 adapter (CEX2C or CEX2A)

The Crypto Express2 adapter is an asynchronous cryptographic coprocessor or accelerator. The adapter contains two cryptographic engines that can be independently configured as a coprocessor (CEX2C) or as an accelerator (CEX2A). It is available on the IBM System z9 Enterprise Class, IBM System z9 Business Class, IBM System z10 Enterprise Class, and IBM System z10 Business Class.

### CP Assist for Cryptographic Functions (CPACF)

CPACF is a set of cryptographic instructions available on all CPs. Use of the CPACF instructions provides improved performance. The SHA-1, SHA-2, and SHA-3 algorithms are always available on the servers where the algorithm is supported. Additional algorithms are available with the appropriate enablement. For more information, see "Server hardware" on page 45.

CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement, feature 3863, provides for clear key DES and TDES instructions. On the z9 EC / z9 BC and later systems, this feature includes clear key AES for 128-bit keys. On z10 EC / z10 BC and later systems, this feature also includes clear key AES for 192-bit and 256-bit keys. On z15 and later systems, this feature also includes ECC algorithm for P-256, P-384, P-521, Ed25519, and Ed448.

## Server hardware

This topic describes the servers on which the cryptographic hardware features are available.

### Regional cryptographic server

Regional cryptographic servers are network-attached, stand-alone devices or dedicated Linux LPARs that perform geography-specific cryptography. Later generations of these servers add international algorithm support. These servers are secure key hardware security modules (HSMs) that operate similar to IBM's PKCS #11 secure coprocessors (CEX*n*P). They are marketed and serviced by third-party vendors. Currently, the only geography-specific cryptography that is supported by these devices is the Chinese SM*x* family of algorithms. Secure keys are stored in the TKDS, protected by the Regional Cryptography Server Master Key (RCS-MK).

The network-attached, stand-alone devices require no particular zSeries hardware, but do require communicating with z/OS V1R13 or later and ICSF FMID HCR77B1 or later. ICSF communicates with these devices using TCP/IP, with optional TLS protection. The Linux LPARs require IBM z13 or later hardware. ICSF communicates with the Linux LPARs using TCP/IP, with TLS protection required.

Once configured and online, ICSF makes the algorithms that are offered by these devices available as PKCS #11 vendor-defined extensions.

- For information on configuring these devices, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.
- For information on the algorithms offered, see *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications* and *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

### IBM z15

The IBM z15 provides constraint relief and addresses various customer demands. It has several cryptographic features.

- CP Assist for Cryptographic Functions is implemented on every processor. SHA-1, SHA-2, and SHA-3 secure hashing and SHAKE extendable output functions are directly available to application programs.

- Feature code 3863, CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement - enables DES, TDES, and AES instructions on all CPs. This feature also includes clear key ECC algorithm for NIST and Edwards curves.
- Feature code 0899, Crypto Express7 adapter - optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM z15 can support a maximum of 60 adapters and the IBM z15 Model T02 can support a maximum of 40 adapters. Each feature code has one hardware adapter which can be configured as a CCA coprocessor, a PKCS #11 coprocessor, or an accelerator.
- Feature code 0898, Crypto Express7 adapter - optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM z15 can support a maximum of 60 adapters and the IBM z15 Model T02 can support a maximum of 40 adapters. Each feature code has two hardware adapters and each can be configured as a CCA coprocessor, a PKCS #11 coprocessor, or an accelerator.
- Feature code 0893, Crypto Express6 adapter - optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM z15 can support a maximum of 16 adapters. Each adapter code has one hardware adapter which can be configured as a CCA coprocessor, a PKCS #11 coprocessor, or an accelerator.
- Feature code 0890, Crypto Express5 adapter - optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM z15 can support a maximum of 16 adapters. Each feature code has one hardware feature which can be configured as a CCA coprocessor, a PKCS #11 coprocessor, or an accelerator.

**Note:** The IBM z15 can have at most 60 Crypto Express adapters installed.

## IBM z14 and IBM z14 ZR1

The IBM z14 and IBM z14 ZR1 provides constraint relief and addresses various customer demands. It has several cryptographic features.

- CP Assist for Cryptographic Functions is implemented on every processor. SHA-1, SHA-2, and SHA-3 secure hashing and SHAKE extendable output functions are directly available to application programs.
- Feature code 3863, CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement - enables DES, TDES, and AES instructions on all CPs.
- Feature code 0893, Crypto Express6 adapter - optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM z14 and IBM z14 ZR1 can support a maximum of 16 adapters. Each adapter code has one hardware adapter which can be configured as a CCA coprocessor, a PKCS #11 coprocessor, or an accelerator.
- Feature code 0890, Crypto Express5 adapter - optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM z14 and IBM z14 ZR1 can support a maximum of 16 adapters. Each feature code has one hardware feature which can be configured as a CCA coprocessor, a PKCS #11 coprocessor, or an accelerator.

## IBM z13 and IBM z13s
The IBM z13 and IBM z13s provide constraint relief and addresses various customer demands. It has several cryptographic features.

- CP Assist for Cryptographic Functions is implemented on every processor. SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 secure hashing is directly available to application programs.
- Feature code 3863, CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement - enables clear key DES and TDES instructions on all CPs. AES 128-bit, AES 192-bit and AES 256-bit support is also available.
- Feature code 0890, Crypto Express5 adapter - optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM z13s can support a maximum of 16 adapters. Each feature code has one hardware feature which can be configured as a CCA coprocessor, a PKCS #11 coprocessor, or an accelerator.

## IBM zEnterprise EC12 (zEC12) and IBM zEnterprise BC12 (zBC12)

The IBM zEnterprise EC12 and IBM zEnterprise BC12 provide constraint relief and addresses various customer demands. It has several cryptographic features.

- CP Assist for Cryptographic Functions is implemented on every processor. SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 secure hashing is directly available to application programs.
- Feature code 3863, CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement - enables clear key DES and TDES instructions on all CPs. AES 128-bit, AES 192-bit and AES 256-bit support is also available.
- Feature code 0864, Crypto Express3 adapter - optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM zEnterprise EC12 can support a maximum of 8 adapters. Each feature code has two coprocessors/accelerators.
- Feature code 0865, Crypto Express4 adapter - optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM zEnterprise EC12 can support a maximum of 16 adapters. Each feature code has one hardware feature which can be configured as a CCA coprocessor, a PKCS #11 coprocessor, or an accelerator.

## IBM zEnterprise 196 (z196) and IBM zEnterprise 114 (z114)

The IBM zEnterprise 196 and IBM zEnterprise 114 provide constraint relief and addresses various customer demands. It has several cryptographic features.

- CP Assist for Cryptographic Functions is implemented on every processor. SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 secure hashing is directly available to application programs.
- Feature code 3863, CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement – enables clear key DES and TDES instructions on all CPs. AES 128-bit, AES 192-bit and AES 256-bit support is also available.
- Feature code 0864, Crypto Express3 adapter – optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM zEnterprise 196 and IBM zEnterprise 114 can support a maximum of 8 adapters. Each feature code has two coprocessors/accelerators.

## IBM System z10 Enterprise Class (z10EC) and IBM System z10 Business Class (z10 BC)

The IBM System z10 Enterprise Class and IBM System z10 Business Class provide constraint relief and addresses various customer demands. It has several cryptographic features.

- CP Assist for Cryptographic Functions is implemented on every processor. SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 secure hashing is directly available to application programs.
- Feature code 3863, CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement – enables clear key DES and TDES instructions on all CPs. AES 128-bit, AES 192-bit and AES 256-bit support is also available.
- Feature code 0863, Crypto Express2 adapter – optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The z10 EC and z10 BC can support a maximum of 8 adapters. Each feature code has two coprocessors/accelerators.
- Feature code 0864, Crypto Express3 adapter – optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The z10 EC and z10 BC can support a maximum of 8 adapters. Each feature code has two coprocessors/accelerators.

## IBM System z9 Enterprise Class (z9 EC) and IBM System z9 Business Class (z9 BC)

The IBM System z9 Enterprise Class (z9 EC) and IBM System z9 BC provide constraint relief and addresses various customer demands. It has several cryptographic features.

- CP Assist for Cryptographic Functions is implemented on every processor. SHA-1, SHA-224 and SHA-256 secure hashing is directly available to application programs.

- Feature code 3863, CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement – enables clear key DES and TDES instructions on all CPs. In addition, ICSF supports hardware implementation of AES 128-bit keys and software implementation of AES 192-bit and AES 256-bit key lengths.
- Feature code 0863, Crypto Express2 adapter – optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM System z9 BC can support a maximum of 8 adapters. Each feature code has two coprocessors/accelerators.

# Configuring Servers and Cryptographic Processors

There is only LPAR mode on your servers. You can divide your processor complex into PR/SM logical partitions. When you create logical partitions on your processor complex, you use the usage domain index on the Support Element Customize Image Profile page only if you have (or plan to add) cryptographic hardware features.

The DOMAIN parameter in the ICSF installation options data set is optional. The number that is specified for the usage domain index must correspond to the domain number you specified with the DOMAIN(n) keyword in the installation options data set – if you specified one. The DOMAIN keyword is required if more than one domain is specified as the usage domain on the PR/SM panels.

A cryptographic feature can be configured and shared across multiple partitions.

**Note:** The domain assigned to the TKE host LPAR must be unique if TKE is to control all the cryptographic features cards in the environment. No other LPAR can use the domain assigned to the TKE host.

The maximum number of LPARs depends on your server. With IBM z13, z13s, and later systems, the maximum number of usage domains matches the maximum number of LPARs available on the server. A usage domain can be configured to be unique to one LPAR or assigned to different LPARs accessing different cryptographic features. This is illustrated by LPAR 1 and LPAR 3 in Figure 12 on page 48. They are both assigned to usage domain 0 but on two different CEX3As.
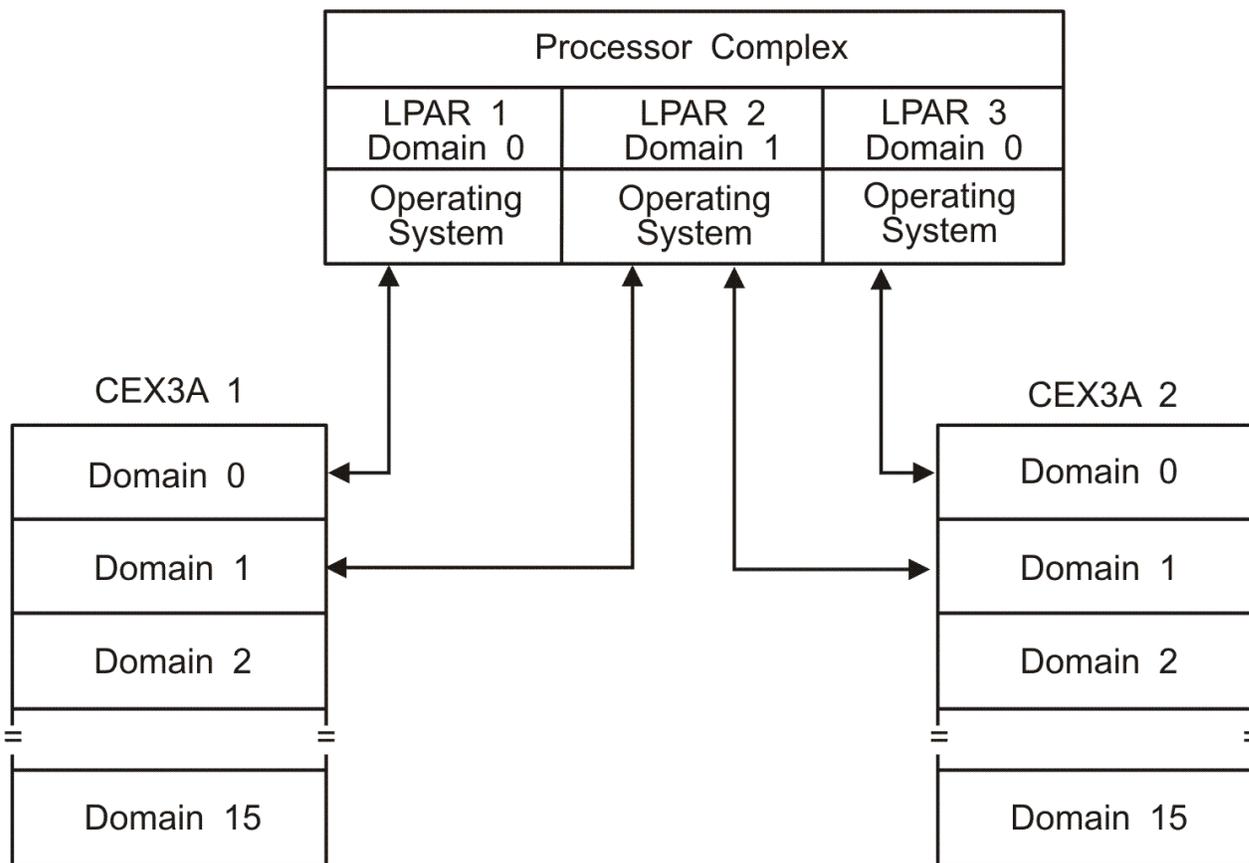


Figure 12. Two Crypto CEX3As on a processor complex running in LPAR mode

The example in Figure 12 on page 48 shows that LPAR 2 has assigned access to Domain 1 on both CEX3A 1 and CEX3A 2. If you were to add another feature to LPAR 2, Domain 1 on the new feature' would also be assigned.
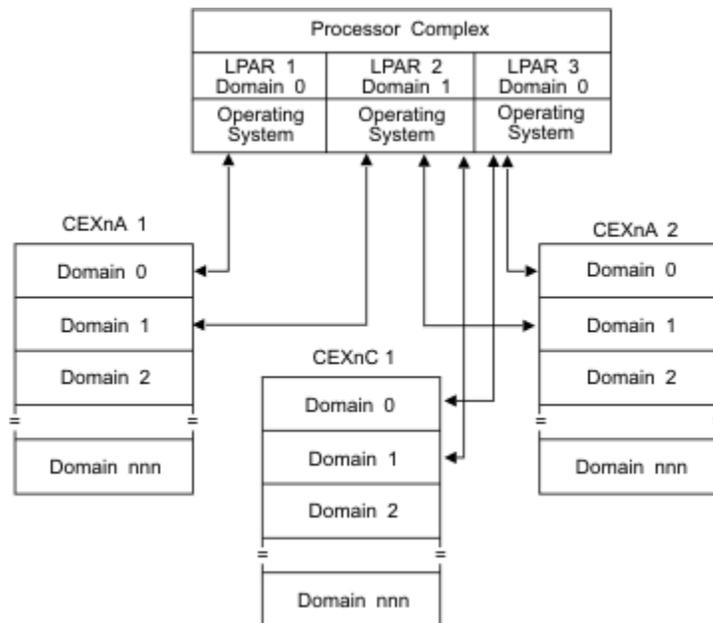


*Figure 13. Multiple Crypto coprocessors on a complex running in LPAR mode*

The example in Figure 12 on page 48 shows that LPAR 2 has assigned access to Domain 1 on CEX3A 1, CEX3A 2, and CEX3A 1. LPAR 3 has assigned access to Domain 0 on CEX3A and CEX3A 1.

# Security

In reviewing your installation security plan before installing ICSF, consider these points:

- **Controlling access to disk copies of the CKDS**

  You should determine which users and applications should have access to each copy of the CKDS on your system.

  **Note:** The in-storage copy of the CKDS can be accessed only through ICSF functions such as callable services or the ICSF panels. To protect the in-storage copy of the CKDS, control who can use these services.

- **Controlling access to the PKDS**

  You should determine which users and applications should have access to the PKDS on your system.

  **Note:** The in-storage copy of the PKDS can be accessed only through ICSF functions such as callable services or the ICSF panels. To protect the in-storage copy of the PKDS, control who can use these services.

- **Controlling access to the Key Generator Utility Program (KGUP)**

  Anyone who is running the KGUP can read and change an unprotected CKDS. To prevent unauthorized persons from using the KGUP, store the program in an APF-authorized library that is protected by the Security Server (RACF). KGUP is also protected by CSFSERV(CSFKGUP).

- **Controlling access to services**

  Users of the Security Server (RACF) can use the CSFSERV class to perform access checking and auditing of services. The CSFSERV class also protects critical administrative TSO panel utilities, such as changing the master key and refreshing a key data set. The audit records that are produced by these checks are SMF type 80 records.

In addition, the cryptographic coprocessors have access control for services and some panel utilities. These controls can be managed using the TKE workstation.

- **Controlling access to key material**

  Users of the Security Server (RACF) can use the CSFKEYS and CRYPTOZ classes to perform access checking and auditing of key material. The audit records that are produced by these check are SMF type 80 records.

  Key stores collectively refers to the Cryptographic Key Data Set (CKDS) and Public Key Data Set (PKDS) and their contents. ICSF provides additional security options for key material that are referred to as Key Store Policy. The audit records that are produced by these checks are SMF type 82 records.

- **Key token authorization checking controls**

  ICSF will verify, when an application passes a callable service a key token instead of a key label, that the user has authority to the secure token. ICSF does this by identifying key labels associated with the key token so that a SAF authorization check (which depends on key labels) can be carried out against profiles in the CSFKEYS class. An audit record is produced when the key token is not authorized.

- **Default key label checking controls**

  Administrators can define what happens when a key token is passed to a callable service instead of a key label.

- **Duplicate key token checking controls**

  ICSF prevents applications and the Key Generator Utility Program (KGUP) from storing duplicate tokens in a CKDS or PKDS.

- **Granular key label access controls**

  Administrators can raise the level of access authority required to create, write to, or delete a key label.

- **Symmetric key label export controls**

  Administrators can raise the level of access authority required to export a symmetric key (transfer it from encryption under a master key to encryption under an application-supplied RSA public key) when an application calls the Symmetric Key Export callable service (CSNDSYX or CSNFSYX). In this case, a SAF authorization check is performed against profiles in the XCSFKEY class rather than the CSFKEYS class.

- **PKA key management extensions controls**

  Administrators can set additional restrictions on how keys can be used. These additional restrictions are specified in the ICSF segment of CSFKEYS (or XCSFKEY) profiles. Using the ICSF segment of profiles in these classes, you can:

  - Specify that asymmetric keys covered by the profile cannot be used for secure export or import operations.
  - Specify that asymmetric keys covered by the profile cannot be used in the handshake operations performed by the Digital Signature Generate (CSNDDSG and CSNFDSG), Digital Signature Verify (CSNDDSV and CSNFDSV), PKA Encrypt (CSNDPKE and CSNFPKE), and PKA Decrypt (CSNDPKD and CSNFPKD) callable services.
  - Specify whether symmetric keys covered by the profile can be exported using the Symmetric Key Export callable service (CSNDSYX or CSNFSYX). If allowing the symmetric keys covered by the profile to be exported, you can specify which asymmetric keys can be used to perform the export operation. You can specify this by supplying a list of labels of RSA keys in the PKDS, or a list of certificates in either a PKCS #11 token, or a SAF key ring.

- **Key store record archiving**

  ICSF administrators determine which records in the key data sets are unused and can be deleted. Administrators can mark key data set records are archived. These records cannot be used by an application unless the administrator allows their use through an optional control. An audit record is produced for the use of an archived record. An optional joblog message can be issued for the first attempt that an archived record is used.

- **Key material validity**

  ICSF administrators can define a period when the key material of a key data set record can be used by applications. Any attempt to use the record outside of the validity dates causes the service request to fail and produce an audit record.

  You should familiarize yourself with the controls that you can enable and decide on the Key Store Policy that is best for your installation.

- **Enforcing PIN block restrictions**

  Requirements in the ANSI X9.8 PIN standard intended to guard against PIN block attacks can be implemented by enabling certain access control points on the Crypto Express3 Coprocessor using the optional Trusted Key Entry (TKE) workstation. These access control points are:

  – ANSI X9.8 PIN - Enforce PIN block restrictions
  – ANSI X9.8 PIN - Allow modification of PAN
  – ANSI X9.8 PIN - Allow only ANSI PIN blocks

  When enabled, these access control points limit the capabilities of the following callable services:

  – Clear PIN Generate Alternate (CSNBCPA and CSNECPA)
  – Encrypted PIN Translate (CSNBPTR and CSNEPTR)
  – Secure Messaging for PINs (CSNBSPN and CSNESPN)

- **Scheduling changes for cryptographic keys**

  To reduce the possibility of exposing a key value, you may want to change the value of cryptographic keys, including master keys, from time to time:

  – You can use the ICSF panels to change the DES, AES, ECC and RSA master keys.
  – If you have an optional Trusted Key Entry (TKE) workstation installed, you can use it to change DES, AES, ECC and RSA master keys on all cryptographic coprocessors.
  – You can use KGUP or the ICSF panel to change the CKDS.
  – You can develop applications that use the dynamic CKDS update callable services to change both the in-storage and DASD copies of the CKDS.
  – You can develop applications that use the dynamic PKDS update callable services to change both the in-storage and DASD copies of the PKDS.

  You can perform all of these operations without interrupting cryptographic functions.

- **Allowing or preventing clear cryptographic keys**

  With ICSF, keys exist in the clear only in these cases:

  – if you specifically allow special secure mode, applications can use the Clear PIN Generate, Secure Key Import, Secure Key Import2, and Multiple Secure Key Import callable services. Access control points for these services must be enabled.
  – If ICSF is not in special secure mode, most keys in the system are encrypted except DATA keys that a user may enter through the use of the clear key import callable service. CLRAES and CLRDES keys are also not encrypted.

    **Note:** The clear key import callable service is equivalent to the PCF EMK macro.
  – The encode callable service can use a clear key to encipher data.
  – If you use the Master Key Entry panels to enter the key parts of a master key, the key parts appear briefly in the clear in host storage.
  – Clear keys are used to provide improved performance for the DES, TDES and AES algorithms. Symmetric key encrypt and decrypt services (CSNBSYD and CSNBSYE) are available on all CP's.

- **Sending cryptographic keys to other installations**

  To eliminate the need to have a courier deliver clear keys between installations, you can use either or both of these options:

- – AES and DES transport keys to encrypt CCA keys or TR-31 key blocks for network distribution.
- – The receiving installation's RSA public key to encrypt a DES or AES DATA key prior to electronic distribution.

Both of these methods make key distribution more secure.

- • **Controlling access to the disk copies**

  You should determine which users and applications should have access to each DASD copy of the CKDS, PKDS and TKDS on your system.

- • **SMF records generated by ICSF**

  ICSF generates type 82 records in the SMF data set when these conditions occur:

  - – ICSF starts.
  - – A cryptographic processor is configured online or offline.
  - – When the in-storage CKDS is refreshed.
  - – When the in-storage PKDS is refreshed.
  - – You use the ICSF panels to process an operational key loaded using the TKE workstation.
  - – When an application uses any of the dynamic services that updates the CKDS.
  - – When an application uses any of the dynamic services that updates the PKDS.
  - – When you use the Master Key Entry panels to enter a master key in the cryptographic coprocessor.
  - – When you create or delete a retained key on a coprocessor.
  - – When you use the TKE workstation to communicate with cryptographic coprocessors.
  - – To capture measurements of timing and configuration for the cryptographic feature coprocessor or accelerator.
  - – When ICSF joins or leaves the ICSF sysplex group.
  - – When you use the Trusted Block Create callable service to create or activate a trusted block.
  - – When you use the PKCS #11 token management callable services to create or delete a token or object or to modify an attribute value of an object.
  - – When the security administrator has indicated that duplicate key tokens must be identified.
  - – When a callable service fails a key store policy check.
  - – When TKE audit records are processed.
  - – When CKDS, PKDS, or TKDS metadata is changed.
  - – When an archived or inactive CKDS, PKDS, or TKDS record is referenced.
  - – Key lifecycle events are audited.
  - – Key usage events are audited.
  - – Cryptographic usage statistics are recorded.
  - – Compliance warning event information is recorded.
  - – A master key event resulted in a new master key value being promoted to current.

  The SMF record type 82 subtype indicates the type of event that caused ICSF to write the SMF record. For subtypes that log state changes, the SMF record will contain additional audit information about the server user and, optionally, the end user.

  You can also use the Security Server (RACF) or an equivalent product to record attempts to use protected cryptographic keys or functions.

- • **Recording and formatting type 82 SMF records in a report**

  Sample jobs are available (in SYS1.SAMPLIB) to assist in the recording and formatting of type 82 SMF data:

  - – **CSFSMFJ** - JCL that executes the code to dump and format SMF type 82 records for ICSF. Before executing the JOB step, you need to make modifications to the JCL (see the prologue in the sample

for specific instructions). After the JCL has been modified, terminate SMF recording of the currently active dump dataset (by issuing I SMF) to allow for the unloading of SMF records. After SMF recording has been terminated, execute the JCL. The output goes into the held queue.

– **CSFSMFR** - An EXEC that formats the SMF records into a readable report.

• **Recording and formatting type 80 SMF records in a report**

RACF provides support to log type 80 SMF records when a user attempts to access an ICSF service, utility, or key label when a profile is defined for the service, utility or label. See the *z/OS Security Server RACF Auditor's Guide* for guidance on how to activate this logging and to format the type 80 SMF records.

# Operating considerations

Before operating a computing system that has ICSF installed, you should consider certain items.

## ICSF initialization options

Your system operator can use the START and STOP operator commands to start and stop ICSF. Also, your system programmer can set up different sets of options such as a PARMLIB member that the operator can specify on the START command. This enables you to set ICSF up to run differently at different times. For more information, see "Using options to tailor ICSF" on page 19.

## LPAR considerations

You can divide your processor complex into PR/SM logical partitions (LPARs) by assigning crypto CP master key registers or domains to each LPAR. When running in LPAR mode, use system symbols in the installation options data set to define different domains. You can assign one or more domains to an LPAR.

The DOMAIN parameter is an optional parameter in the installation options data set. It is required if more than one domain is specified as the usage domain on the PR/SM panels or if running in native mode. If you assign multiple domains to an LPAR, you can have separate master keys for different purposes. For instance, you might have one master key for production operations and another master key for test operations.

You are able to use the same domain in more than one LPAR as long as you're not sharing the same cryptographic feature coprocessor.

## Link Pack Area (LPA) considerations

ICSF uses dynamic LPA to load the pre-PC routines, CICS-related routines, and other modules into ECSA and CSA. The dynamic LPA load will occur the first time that ICSF is started within an IPL, and the modules will persist across subsequent restarts of ICSF.

# Appendix A. Standards

IBM cryptographic coprocessor features and ICSF support cryptographic algorithms and techniques from International and geographical standards organizations.

The following organizations publish standards that IBM cryptographic coprocessor features and ICSF provides support for (at least in part):

**ANSI**
American National Standards Institute.

**BSI**
Federal Office for Information Security in Germany.

**DK**
(formerly ZKA) German Banking Industry Committee.

**EMV**
Europay, MasterCard, Visa (American Express, Japan Credit Bureau (JCB), Discover, UnionPay).

**ISO**
International Organization for Standardization.

**IETF**
Internet Engineering Task Force.

> **RFCs**
> Technical documents describing computer networking, protocols, procedures, programs, and concepts.

**NIST**
National Institute of Standards and Technology.

> **FIPS**
> Federal Information Processing Standards. U.S. federal government standards for Computer Systems.

**OASIS**
Cryptographic Token Interface Standard (v2.40).

**PCI**
Payment Card Industry Security Standards Council.

**RSA Laboratories**
Cryptographic Token Interface Standard (up to v2.20).

The following is a summary of the cryptographic algorithms and techniques supported by the IBM cryptographic coprocessor features and ICSF for each of the above organizations:

**ANSI – ASC X9**
Accredited Standards Committee X9.

**ANSI X9.8**
Personal Identification Number (PIN) Management and Security.

The following access control points (ACPs) can be used to guard against PIN block attacks:

- ANSI X9.8 PIN - Enforce PIN block restrictions.
- ANSI X9.8 PIN - Allow modification of PAN.
- ANSI X9.8 PIN - Allow only ANSI PIN blocks.
- ANSI X9.8 PIN - Use stored decimalization tables only.

When enabled, these ACPs limit the capabilities of the following callable services:

- Clear PIN Generate Alternate (CSNBCPA and CSNECPA).

- Encrypted PIN Generate (CSNBEPG and CSNEEPG).
- Encrypted PIN Translate (CSNBPTR and CSNEPTR).
- Encrypted PIN Verify (CSNBPVR and CSNEPVR).
- Secure Messaging for PINs (CSNBSPN and CSNESPN).

**ANSI X9.9**

Financial Institution Message Authentication (Wholesale).

For ANSI standard X9.9, option 1, ICSF can use either a single-length MAC key or a single-length DATA key. The callable services affected are:

- MAC Generate (CSNBMGN or CSNBMGN1 and CSNEMGN or CSNEMGN1).
- MAC Verify (CSNBMVR or CSNBMVR1 and CSNEMVR or CSNEMVR1).

ICSF provides a utility to edit ASCII text string according to rules defined by ANSI X9.9-4:

**X9.9 Data Editing (CSNB9ED)**

CR and LF replaced with single-space, lowercase to uppercase, deletes certain characters, deletes leading spaces, replaces multiple spaces with single space.

**ANSI X9.19**

Optional Double-MAC Procedure.

For ANSI standard X9.19, ICSF uses a double-length MAC generation key. The following callable services support this method:

- MAC Generate (CSNBMGN or CSNBMGN1 and CSNEMGN or CSNEMGN1).
- MAC Verify (CSNBMVR or CSNBMVR1 and CSNEMVR or CSNEMVR1).

**ANSI X9.23**

Encryption of Wholesale Financial Messages.

The ANS X9.23 standard defines an enhancement to the basic cipher block chaining (CBC) mode of NIST SP 800-38A. The callable services affected are:

- Ciphertext Translate2 (CSNBCTT2, CSNBCTT3, CSNECTT2, CSNECTT3).
- Decipher (CSNBDEC or CSNBDEC1 and CSNEDEC or CSNEDEC1).
- Encipher (CSNBENC or CSNBENC1 and CSNEENC or CSNEENC1).
- Symmetric Key Decipher (CSNBSYD or CSNBSYD1 and CSNESYD or CSNESYD1).
- Symmetric Key Encipher (CSNBSYE or CSNBSYE1 and CSNESYE or CSNESYE1).

**ANSI X9.24**

Retail Financial Services Symmetric Key Management.

Derived Unique Key Per Transaction (for double-length PIN keys). The following callable service supports this method:

- Unique Key Derive (CSFBUKD and CSFEUKD).

Defines key wrapping techniques using CBC mode to prevent attacks against key bundling and keys wrapped by other keys.

- ICSF initialization option, DEFAULTWRAP, is used to specify which method (original or enhanced) is the default. The enhanced method specifies the ANSI X9.24 compliant wrapping method.
- Some ICSF services contain rule array keywords that allow the DEFAULTWRAP method to be overridden. The callable services affected are:
  - Ciphertext Translate2 (CSNBCTT2, CSNBCTT3, CSNECTT2, CSNECTT3).
  - Control Vector Translate (CSNBCVT and CSNECVT).
  - Data Key Export (CSNBDKX and CSNEDKX).
  - Data Key Import (CSNBDKM and CSNEDKM).

- Diversified Key Generate (CSNBDKG and CSNEDKG).
- Key Export (CSNBKEX and CSNEKEX).
- Key Generate (CSNBKGN and CSNEKGN).
- Key Import (CSNBKIM and CSNEKIM).
- Key Token Build (CSNBKTB and CSNEKTB).
- Key Translate (CSNBKTR and CSNEKTR).
- Multiple Clear Key Import (CSNBCKM and CSNECKM).
- Multiple Secure Key Import (CSNBSKM and CSNESKM).
- Prohibit Export (CSNBPEX and CSNEPEX).
- Prohibit Export Extended (CSNBPEXX and CSNEPEXX).
- Remote Key Export (CSNDRKX and CSNFRKX).
- Secure Key Import (CSNBSKI and CSNESKI).
- Symmetric Key Generate (CSNDSYG and CSNFSYG).
- Symmetric Key Import (CSNDSYI and CSNFSYI).
- TR-31 Export (CSNBT31X and CSNET31X).
- TR-31 Import (CSNBT31I and CSNET31I).

- A key must not be wrapped by a key weaker than itself. ICSF provides ACPs to prevent this:

  - Prohibit weak wrapping - Transport keys.
  - Prohibit weak wrapping - Master keys.
  - Warn when weak wrap - Transport keys.
  - Warn when weak wrap - Master keys.

**ANSI TR-31**

Interoperable Secure Key Exchange Key Block Specification for Symmetric Algorithms.

Specifies a method consistent with the requirements of ANSI X9.24 for the secure exchange of keys and other sensitive data between two devices that share a symmetric key exchange key. The following callable services support this method:

- TR-31 Export (CSNBT31X and CSNET31X).
- TR-31 Import (CSNBT31I and CSNET31I).
- TR-31 Parse (CSNBT31P and CSNET31P).
- TR-31 Optional Data Read (CSNBT31R and CSNET31R).
- TR-31 Optional Data Build (CSNBT31O and CSNET31O).

DES/TDES and AES keys can be transported in TR-31 key blocks.

**ANSI TR-34**

Interoperable Method for Distribution of Symmetric Keys using Asymmetric Techniques.

Specifies a protocol for distribution of symmetric keys from a Key Distribution Host (KDH) to one or more remote Key Receiving Devices (KRDs), such as ATMs, using public-key techniques. It is an implementation of the Unilateral Key Transport Method defined in ANS X9.24-2. The following callable services support this method:

- TR-34 Bind-Begin (CSNDT34B and CSNFT34B).
- TR-34 Bind-Complete (CSNDT34C and CSNFT34C).
- TR-34 Key Distribution (CSNDT34D and CSNFT34D).
- TR-34 Key Receive (CSNDT34R and CSNFT34R).

**VISA**

Integrated Circuit Card Specification.

ICSF supports the PIN change algorithms specified in the VISA Integrated Circuit Card Specification. The following callable service supports this method:

- PIN Change/Unblock (CSNBPCU and CSNEPCU).
  - Keywords are used to specify each PIN Block Format: AMEXPCU1, AMEXPCU2, VISAPCU1, and VISAPCU2.

**EMV**

Integrated Circuit Card Specifications for Payment Systems.

ICSF provides a comprehensive set of key management services that allow you to create, generate, derive, import, and export keys needed for EMV online authorization processing.

ICSF provides services you can use in secure communications with EMV smart cards.

EMV ICC integrated circuit card applications can use the following callable services:

- Derive ICC MK (CSNBDCM and CSNEDCM).
- Derive Session Key (CSNBDSK and CSNEDSK).
- Diversified Key Generate (CSNBDKG and CSNEDKG).
- EMV Scripting Service (CSNBESC and CSNEESC).
- EMV Transaction Service (CSNBEAC and CSNEEAC).
- EMV Verification Functions (CSNBEVF and CSNEEVF).
- Generate Issuer MK (CSNBGIM and CSNEGIM).
- Secure Messaging for Keys (CSNBSKY and CSNESKY).
- Secure Messaging for PINs (CSNBSPN and CSNESPN).

The Diversified Key Generate (CSNBDKG and CSNEDKG) service supports the EMV2000 key derivation methods.

**ISO 9796**

Information Technology - Security Techniques - Digital Signature Scheme Giving Message Recovery.

The Digital Signature Generate (CSNDDSG and CSNFDSG) and Digital Signature Verify (CSNDDSV and CSNFDSV) services support the following digital signature schemes:

- ANSI X9.62 (ECDSA).
- ANSI X9.31 (RSA).
- ISO 9796-1 (RSA).
- RSA DSI PKCS 1.0 and 1.1 (RSA).
- Padding on the left with zeros (RSA).

**ISO 9564**

Personal Identification Number Management and Security Part 1 - PIN Protection.

ICSF PIN services support the following ISO PIN block formats:

- ISO PIN block format 0.
- ISO PIN block format 1.
- ISO PIN block format 2.
- ISO PIN block format 3.

**ISO 16609**

Financial services - Requirements for message authentication using symmetric techniques.

ICSF supports use of the ISO 16609 algorithm with a double-length MAC or a double- length DATA key and the message text using TDES and CBC mode. The following callable services support these methods:

- MAC Generate (CSNBMGN or CSNBMGN1 and CSNEMGN or CSNEMGN1).

- MAC Verify (CSNBMVR or CSNBMVR1 and CSNEMVR or CSNEMVR1).

**NIST SP 800-38A**
Recommendation for Block Cipher Modes of Operations.

- Electronic Codebook (ECB).
- Cipher Block Chaining (CBC).
- Cipher Feedback (CFB).
- Output Feedback (OFB).
- Counter (CTR).

**NIST SP 800-38B**
Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication.

**NIST and FIPS 140-2**
The z/OS PKCS #11 services are designed to meet FIPS 140-2 Level 1 criteria and can be configured to operate in compliance with FIPS 140-2 specifications.

The cryptographic coprocessors are designed to meet FIPS 140-2 Level 4 criteria.

**FIPS 140 Mode Processing**
ICSF Installation Option for configuring FIPS Mode Processing.

FIPSMODE(YES or COMPAT or NO,FAIL(fail-option))

**YES**
Any application using the PKCS #11 services is forced to use those services in a FIPS-compliant manner.

**COMPAT**
This mode is intended for installations where only certain z/OS PKCS #11 applications must comply with the FIPS 140-2 standard while other applications do not. Resource profiles are defined in the CRYPTOZ class and Boolean key attributes are used to exempt tokens from FIPS compliance.

**NO**
FIPS no-enforcement mode. FIPS restrictions are not enforced unless application explicitly requests enforcement.

**FAIL(fail-option)**
Indicates if ICSF should terminate if there is a failure with FIPS processing.

This option only applies to cryptographic coprocessors configured in Enterprise PKCS#11 (EP11) mode.

**FIPS 46-2**
Data Encryption Standard (DES).

**FIPS 180-4**
Secure Hash Standard.

ICSF can use SHA-256 to produce Key Check Values (KCVs) via its Key Test (CSNBKYT and CSNEKYT) and Key Test2 (CSNBKYT2 and CSNEKYT2) callable services.

**FIPS 197**
Advanced Encryption Standard - (AES).

**FIPS 198**
Keyed-Hash Message Authentication Code (HMAC).

**PKCS #11, V2.20 (also known as Cryptoki)**
Cryptographic Token Interface Standard.

ICSF Enterprise PKCS#11 Algorithms Supported (FIPS approved algorithms and sizes only).

Digital Signatures (SHA hashing provided by CPACF):

- DSA: Key sizes 1024 – 2048 bit.

- ECDSA, Prime Curves:
  - NIST – 192, 224, 256, 384, 521 bit.
  - Brainpool – 160, 192, 224, 320, 384, 512.
- RSA, Key sizes 1024 – 4096 bit.
  - PKCS #1v1.5.
  - PKCS #1 v2.1 PSS.

Encryption:

- RSA, PKCS #1 v1.5: Key sizes 1024 – 4096 bit.
- TDES: 3-key only, ECB and CBC modes, with or without PKCS PAD.
- AES: Key sizes 128, 192, 256 bit, ECB and CBC modes.

Standard PKCS#11 key wrapping mechanisms:

- Wrap private or secret keys using symmetric encryption:
  - TDES: 3-key only, CBC mode with PKCS PAD.
  - AES: key sizes 128, 192, 256 bit, CBC mode with PKCS PAD.
  - Wrapping secret keys with symmetric encryption.
- Wrap secret keys using asymmetric encryption:
  - RSA, PKCS #1 v1.5.

Attribute Bound Wrap - Custom mechanism:

- Key usage attributes bound with key material:
  - Cannot be wrapped with standard PKCS#11 method.
- Cryptogram signed with digital signature or HMAC key:
  - Signature verified during unwrap.

HMAC:

- SHA1, SHA224, SHA256, SHA384, SHA512:
  - Generic Secret keys - Minimum key size is half of the SHA output size.

Random Number Generate (RNG):

- The PKCS #11 Pseudo-random function (CSFPPRF and CSFPPRF6) service may be called with the PRNGFIPS mechanism which specifies that pseudo-random bytes must be generated consistent with requirements from NIST SP 800-90.

Diffie-Hellman (DH) and Elliptic Curve Diffie-Hellman (ECDH):

- DH and ECDH secure key-pair generate.
- DH and ECDH secure key derivation:
  - Derived secret key is always clear - optimal performance for session keys.
- DH and DSA Domain Parameter Generate:
  - This is CPU offload only since domain parameters are always public.

**DK AES PIN**

These financial services are based on the PIN methods of and meet requirements specified by the German Banking Industry Committee, Die Deutsche Kreditwirtschaft, also known as DK. The intellectual property rights regarding the methods and specification belongs to the German Banking Industry Committee:

- DK Deterministic PIN Generate (CSNBDDPG and CSNEDDPG).
- DK Migrate PIN (CSNBDMP and CSNEDMP).

- DK PAN Modify in Transaction (CSNBDPMT and CSNEDPMT).
- DK PAN Translate (CSNBDPT and CSNEDPT).
- DK PIN Change (CSNBDPC and CSNEDPC).
- DK PIN Verify (CSNBDPV and CSNEDPV).
- DK PRW Card Number Update (CSNBDPNU and CSNEDPNU).
- DK PRW Card Number Update2 (CSNBDCU2 and CSNEDCU2).
- DK PRW CMAC Generate (CSNBDPCG and CSNEDPCG).
- DK Random PIN Generate (CSNBDRPG and CSNEDRPG).
- DK Random PIN Generate2 (CSNBDRG2 and CSNEDRG2).
- DK Regenerate PRW (CSNBDRP and CSNEDRP).

**PCI DSS**

Payment Card Industry Data Security Standard.

ICSF SMF records along with other IBM products including IBM Security zSecure suite, the Trusted Key Entry (TKE) Workstation, and Enterprise Key Management Foundation (EKMF) offering can be used to address these requirements.

# Appendix B. Summary of callable service support by hardware configuration

The callable services available to your applications depend on the configuration of your server and cryptographic features. The configuration of the cryptographic features depends on U.S. Export Regulations. For information on the configurations available in your country, contact your IBM marketing representative.

## General services

contains a summary of the services that do not use cryptography and have no hardware requirement.

*Table 4. General services*

| Service name | Function |
| --- | --- |
| Character/Nibble Conversion (CSNBXBC and CSNBXCB) | Converts a binary string to a character string or vice versa. |
| Code Conversion (CSNBXEA and CSNBXAE) | Converts EBCDIC data to ASCII data or vice versa. |
| ICSF Query Algorithms (CSFIQA) | Provides information on cryptographic and hashing algorithms. |
| ICSF Query Facility (CSFIQF) | Provides ICSF status information and retrieves information on the CCA and EP11 coprocessors. |
| ICSF Query Facility 2 (CSFIQF2) | Provides information on the cryptographic environment as currently known by ICSF. |
| SAF ACEE Selection (CSFACEE) | Allows the caller to provide the ENVR to use for SAF checks. |
| X9.9 Data Editing (CSNB9ED) | Edits an ASCII text string according to the editing rules of ANSI X9.9-4. |

## CCA services

CCA callable services might have a hardware requirement depending on the cryptographic processing.

contains a summary of the CCA services that do not use cryptography and have no hardware requirement.

*Table 5. Services with no hardware requirement*

| Service name | Function |
| --- | --- |
| Control Vector Generate (CSNBCVG) | Builds a control vector from keywords that are specified as input to the service. |
| Key Data Set List (CSFKDSL) | Generates a list of CKDS, PKDS, or TKDS labels that match a search criteria. |
| Key Data Set Metadata Read (CSFKDMR) | Reads the metadata of a CKDS, PKDS, or TKDS record. |
| Key Data Set Metadata Write (CSFKDMW) | Changes the metadata of a set of CKDS, PKDS, or TKDS records. The in-store copy is updated with the DASD copy. |

*Table 5. Services with no hardware requirement (continued)*

| Service name | Function |
|---|---|
| Key Token Build (CSNBKTB) | Builds an internal or external symmetric key token from the supplied parameters. |
| Key Token Build2 (CSNBKTB2) | Builds an internal or external symmetric key token from the supplied parameters. |
| One–Way Hash Generate (CSNBOWH and CSNBOWH1) | Generates a one-way hash on specified text for the RIPEMD-160 or MD5 algorithm. |
| PKA Key Token Build (CSNDPKB) | Creates an external PKA key token containing a clear, private, or public key or a skeleton token. |
| PKA Public Key Extract (CSNDPKX) | Extracts a PKA public key from a supplied PKA internal or external private key token. |
| TR-31 Optional Data Build (CSNBT31O) | Constructs the optional block data structure for a TR-31 key block. |
| TR-31 Optional Data Read (CSNBT31R) | Obtains lists of the optional block identifiers and optional block lengths, and obtains the data for a particular optional block. |
| TR-31 Parse (CSNBT31P) | Retrieves standard header information from a TR-31 key block without importing the key. |

contains a summary of the CCA services that are used for key store processing. These services do not use a cryptographic coprocessor, but in order to process secure keys in the key stores, an active coprocessor is required.

*Table 6. Services for key store processing*

| Service name | Function |
|---|---|
| CKDS Key Record Create (CSNBKRC) | Adds a key record that contains a key token set to binary zeros to both the in-storage and DASD copies of the CKDS. |
| CKDS Key Record Create2 (CSNBKRC2) | Adds a key record that contains a key token to both the in-storage and DASD copies of the CKDS. |
| CKDS Key Record Delete (CSNBKRD) | Deletes a key record from both the in-storage and DASD copies of the CKDS. |
| CKDS Key Record Read (CSNBKRR) | Copies an internal key token from the in-storage copy of the CKDS to application storage. |
| CKDS Key Record Read2 (CSNBKRR2) | Copies an internal key token from the in-storage copy of the CKDS to application storage. |
| CKDS Key Record Write (CSNBKRW) | Writes an internal key token to the CKDS record specified in the key label parameter. Updates both the in-storage and DASD copies of the CKDS currently in use. |
| CKDS Key Record Write2 (CSNBKRW2) | Writes an internal key token to the CKDS record specified in the key label parameter. Updates both the in-storage and DASD copies of the CKDS currently in use. |
| Key Data Set Record Retrieve (CSFRRT) | Reads a record from the CKDS, PKDS, or TKDS. |

| Table 6. Services for key store processing (continued) | |
|---|---|
| **Service name** | **Function** |
| Key Data Set Update (CSFKDU) | Updates a record in the CKDS, PKDS, or TKDS. |
| PKDS Key Record Create (CSNBKRC) | Writes a new record to the PKDS. |
| PKDS Key Record Delete (CSNBKRD) | Deletes an existing record from the PKDS. |
| PKDS Key Record Read (CSNBKRR) | Reads a record from the PKDS and returns the key token of the record. |
| PKDS Key Record Write (CSNBKRW) | Writes over an existing record in the PKDS. |

The Coordinated KDS Administration (CSFCRC) service uses the cryptographic coprocessors for processing and is available on all servers.

| Table 7. Coordinated KDS Administration service | |
|---|---|
| **Service name** | **Function** |
| Coordinated KDS Administration (CSFCRC) | Performs a KDS refresh or KDS reencipher and change master key operation while allowing applications to update the KDS. In a sysplex environment, this callable service performs a coordinated sysplex-wide refresh or change master key operation from a single ICSF instance. |

Table 8 on page 65 contains a summary of the CCA services that use the CP Assist for Cryptographic Functions (CPACF) instructions.

| Table 8. Services that use the CPACF instructions | |
|---|---|
| **Service name** | **Function** |
| Decode (CSNBDCO) | Decodes an 8-byte string of data by using the electronic code book mode of the DES. |
| Digital Signature Generate (CSNDDSG) | Generates an ECDSA signature for clear or protected key ECC Prime P-256, Prime P-384, and Prime P-521. Generates an EdDSA signature for clear or protected key ECC Ed25519 and Ed244. |
| Digital Signature Verify (CSNDDSV) | Verifies an ECDSA signature for ECC Prime P-256, Prime P-384, and Prime P-521. Verifies an EdDSA signature for ECC Ed25519 and Ed244 |
| Encode (CSNBECO) | Encodes an 8-byte string of data by using the electronic code book mode of the DES. |
| Field Level Decipher (CSNBFLD) | Decrypts database fields, preserving the format of the fields by using the VISA Format Preserving Encryption algorithm. |
| Field Level Encipher (CSNBFLE) | Encrypts database fields, preserving the format of the fields by using the VISA Format Preserving Encryption algorithm. |
| HMAC Generate (CSNBHMG and CSNBHMG1) | Generates a keyed-hashed message authentication code (MAC) for an application supplied text string with a clear key. |

| Table 8. Services that use the CPACF instructions (continued) | |
|---|---|
| **Service name** | **Function** |
| HMAC Verify (CSNBHMV and CSNBHMV1) | Verifies a keyed-hashed message authentication code (MAC) for an application supplied text string with a clear key. |
| MAC Generate2 (CSNBMGN2 and CSNBMGN3) | Generates a keyed-hashed message authentication code (MAC) for an application supplied text string with a clear key. |
| MAC Verify2 (CSNBMVR2 and CSNBMVR3) | Verifies a keyed-hashed message authentication code (MAC) for an application supplied text string with a clear key. |
| MDC Generate (CSNBMDG and CSNBMDG1) | Generates a 128-bit modification detection code (MDC) for a text string that the application program supplies. |
| One–Way Hash Generate (CSNBOWH and CSNBOWH1) for SHA1, SHA2, and SHA3 algorithms | Generates a one-way hash on specified text by using the SHA-1, SHA2, or SHA-3 algorithm. |
| Symmetric Key Decipher (CSNBSYD and CSNBSYD1) | Deciphers data in an address space or a data space. This service is available on machines with triple-DES feature codes. |
| Symmetric Key Encipher (CSNBSYE and CSNBSYE1) | Enciphers data in an address space or a data space. This service is available on machines with triple-DES feature codes. |
| Symmetric MAC Generate (CSNBSMG and CSNBSMG1) | Uses the symmetric MAC generate callable service to generate a 96-bit or 128-bit message authentication code (MAC) for an application-supplied text string using an AES key. |
| Symmetric MAC Verify (CSNBSMV and CSNBSMV1) | Uses the symmetric MAC generate callable service to verify a 96-bit or 128-bit message authentication code (MAC) for an application-supplied text string using an AES key. |

The Random Number Generate (CSNBRNG) and Random Number Generate Long (CSNBRNGL) services use a CCA coprocessor, if available, or the CPACF instruction when there are no active CCA coprocessors available.

| Table 9. Random number generate services | |
|---|---|
| **Service name** | **Function** |
| Random Number Generate (CSNBRNG) and Random Number Generate Long (CSNBRNGL) | Generates an 8-byte random number or a user-specified length random number. The output can be specified in three forms of parity: RANDOM, ODD, and EVEN. |

The following CCA services use a cryptographic accelerator, if available. Accelerators perform clear key RSA operations.

- Digital Signature Verify (CSNDDSV).
- PKA Decrypt (CSNDPKD) with MRP, PKCSOAEP, and ZERO-PAD formatting.
- PKA Encrypt (CSNDPKE) with PKCSOAEP and ZERO-PAD formatting.

Table 10 on page 67 contains a summary of the services that require a CCA coprocessor for processing. The letters represent various configurations.

**Letter A**
z9 EC, z9 BC, z10 EC, and z10 BC with CP Assist for Cryptographic Functions DES/TDES Enablement and CEX2C, or z10 EC and z10 BC with CP Assist for Cryptographic Functions DES/TDES Enablement and CEX3C.

**Letter B**
z114/z196 with CP Assist for Cryptographic Functions DES/TDES Enablement and CEX3C.

**Letter C**
IBM zEnterprise EC12 and BC12 with CP Assist for Cryptographic Functions DES/TDES Enablement and CEX3 and CEX4C.

**Letter D**
IBM z13 and z13s with CP Assist for Cryptographic Functions DES/TDES Enablement and CEX5C.

**Letter E**
IBM z14 and z14 ZR1 with CP Assist for Cryptographic Functions DES/TDES Enablement and CEX5C and CEX6C.

**Letter F**
IBM z15 with CP Assist for Cryptographic Functions DES/TDES Enablement and CEX5C, CEX6C, and CEX7C.

| Table 10. Services that require a CCA coprocessor | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Service Name** | **Function** | **A** | **B** | **C** | **D** | **E** | **F** |
| **Authentication Parameter Generate** | Generates an authentication parameter (AP) and returns it encrypted under a supplied encrypting key. | | | X | X | X | X |
| **CVV Key Combine** | Combines two single-length CCA internal key tokens into 1 double-length CCA key token containing a CVVKEY-A key type. | | X | X | X | X | X |
| **Ciphertext Translate2** | Translates the user-supplied ciphertext from one key to another key. | | | X | X | X | X |
| **Clear Key Import** | Imports a clear DATA key, enciphers it under the master key, and places the result into an internal key token. | X | X | X | X | X | X |
| **Clear PIN Encrypt** | Formats a PIN into a PIN block format (IBM 3621, IBM3624, ISO-0, ISO-1, ISO-2, IBM 4704 encrypting PINPAD, VISA 2, VISA 3, VISA 4, ECI 2, ECI 3) and encrypts the results. | X | X | X | X | X | X |
| **Clear PIN Generate** | Generates a clear personal identification number (PIN), a PIN verification value (PVV), or an offset by using one of these algorithms: • Interbank PIN (INBK-PIN) • IBM 3624 (IBM-PIN or IBM-PINO) • IBM German Bank Pool (GBP-PIN) • VISA PIN validation value (VISA-PVV) | X | X | X | X | X | X |
| **Clear PIN Generate Alternate** | Generates a clear VISA PIN validation value (PVV) from an input encrypted PIN block. | X | X | X | X | X | X |

| Table 10. Services that require a CCA coprocessor (continued) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Service Name** | **Function** | **A** | **B** | **C** | **D** | **E** | **F** |
| **Control Vector Translate** | Changes the control vector that is used to encipher an external key. | X | X | X | X | X | X |
| **Cryptographic Variable Encipher** | Encrypts plaintext by using the Cipher Block Chaining (CBC) method. | X | X | X | X | X | X |
| **Data key Export** | Converts a DATA key from operational form into exportable form. | X | X | X | X | X | X |
| **Data key Import** | Imports an encrypted single-length or double-length DES data key and creates or updates a target internal key token with the master key-enciphered source key. | X | X | X | X | X | X |
| **Decipher** | Deciphers data by using the cipher block chaining mode of the DES. | X | X | X | X | X | X |
| **Derive ICC MK** | Derives ICC master keys from issuer master keys. | X | X | X | X | X | X |
| **Derive Session Key** | Derives session keys from either issuer master keys or ICC master keys. | X | X | X | X | X | X |
| **Digital Signature Generate** | Generates a digital signature by using a supplied hash and a private key. | X | X | X | X | X | X |
| **Digital Signature Verify** | Verifies a digital signature by using the same supplied hash that was used to generate the signature and the public key that corresponds to the private key used to generate the signature. | X | X | X | X | X | X |
| **Diversified Key Generate** | Generates a key based on the key-generating key, the processing method, and the parameter supplied. The control vector of the key-generating key also determines the type of target key that can be generated. | X | X | X | X | X | X |
| **Diversified Key Generate2** | Generates an AES key based on a function of a key-generating key, the process rule, and data that you supply. | | X | X | X | X | X |
| **Diversify Directed Key** | Generates or derive keys using with the DK Direct Key Diversification key scheme. | | | | X | X | X |
| **DK Deterministic PIN Generate** | Generates a PIN and PIN reference value (PRW) by using an AES PIN calculation key. | | X | X | X | X | X |
| **DK Migrate PIN** | Generates the PIN reference value (PRW) for a specified user account. | | X | X | X | X | X |
| **DK PAN Modify in Transaction** | Generates a new PIN reference value (PRW) for an existing PIN when a merger has occurred and the account information has changed. | | X | X | X | X | X |
| **DK PAN Translate** | Creates an encrypted PIN block with the same PIN and a different PAN. | | X | X | X | X | X |

| Table 10. Services that require a CCA coprocessor (continued) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Service Name** | **Function** | **A** | **B** | **C** | **D** | **E** | **F** |
| **DK PIN Change** | Allows a customer to change their PIN to a value of their choosing. | | X | X | X | X | X |
| **DK PIN Verify** | Verifies an ISO-1 format PIN. | | X | X | X | X | X |
| **DK PRW Card Number Update** | Generates a PIN reference value (PRW) when a replacement card is being issued. | | X | X | X | X | X |
| **DK PRW Card Number Update2** | Generates a PIN reference value (PRW) when a replacement card is being issued. | | | | X | X | X |
| **DK PRW CMAC Generate** | Generates a message authentication code (MAC) over specific values that are involved in an account number change transaction. | | X | X | X | X | X |
| **DK Random PIN Generate** | Generates a PIN and a PIN reference value by using the random process. | | X | X | X | X | X |
| **DK Random PIN Generate2** | Generates a PIN and a PIN reference value by using the random process. | | | | X | X | X |
| **DK Regenerate PRW** | Generates a new PIN reference value for a changed account number. | | X | X | X | X | X |
| **ECC Diffie-Hellman** | Creates symmetric key material from a pair of ECC keys by using the Elliptic Curve Diffie-Hellman protocol and the static unified model key agreement scheme or "Z" data (the "secret" material output from D-H process). | | X | X | X | X | X |
| **EMV Scripting Service** | Simplifies EMV scripting. Scripts can be encrypted for confidentiality, MAC'd for integrity, or both. | X | X | X | X | X | X |
| **EMV Transaction Service** | Simplifies ARQC verification and ARPC generation. | X | X | X | X | X | X |
| **EMV Verification Functions** | Provides EMV functions that are used by MasterCard. | X | X | X | X | X | X |
| **Encipher** | Enciphers data by using the cipher block chaining mode of the DES. | X | X | X | X | X | X |
| **Encrypted PIN Generate** | Generates and formats a PIN and encrypts the PIN block. | X | X | X | X | X | X |
| **Encrypted PIN Translate** | Reenciphers a PIN block from one PIN-encrypting key to another and optionally, changes the PIN block format. | X | X | X | X | X | X |
| **Encrypted PIN Translate2** | Reenciphers a PIN block from one PIN-encrypting key to another and optionally, changes the PIN block format. | | | | X | X | X |

| Table 10. Services that require a CCA coprocessor (continued) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Service Name** | **Function** | **A** | **B** | **C** | **D** | **E** | **F** |
| **Encrypted PIN Verify** | Verifies a supplied PIN by using one of these algorithms:<br><br>• Interbank PIN (INBK-PIN)<br>• IBM 3624 (IBM-PIN or IBM-PINO)<br>• IBM German Bank Pool (GBP-PIN)<br>• VISA PIN validation value (VISA-PVV) | X | X | X | X | X | X |
| **Format Preserving Algorithms Decipher** | Decrypts payment card data using FFX algorithms. | | | | | | X |
| **Format Preserving Algorithms Encipher** | Encrypts payment card data using FFX algorithms. | | | | | | X |
| **Format Preserving Algorithms Translate** | Translates payment card data from encryption under one key to encryption under another key using FFX algorithms. | | | | | | X |
| **FPE Decipher** | Decrypts payment card data using Visa Data Secure Platform (Visa DSP) processing. | | | | X | X | X |
| **FPE Encipher** | Encrypts payment card data using Visa Data Secure Platform (Visa DSP) processing. | | | | X | X | X |
| **FPE Translate** | Translates payment card data from encryption under one key to encryption under another key using Visa Data Secure Platform (Visa DSP) processing. | | | | X | X | X |
| **Generate Issuer MK** | Generates issuer master keys and stores the keys in the CKDS. | X | X | X | X | X | X |
| **HMAC Generate** | Generates a keyed-hashed message authentication code (MAC) for a text string that the application program supplies. The MAC is computed by using the FIPS-198 algorithm. | | X | X | X | X | X |
| **HMAC Verify** | Verifies a keyed-hashed message authentication code (MAC) for a text string that the application program supplies. The MAC is computed by using the FIPS-198 algorithm. | | X | X | X | X | X |
| **ICSF Multi-Purpose Service** | Validates the keys in the active CKDS or PKDS. | | | | X | X | X |
| **Key Export** | Converts any key from operational form into exportable form. | X | X | X | X | X | X |
| **Key Generate** | Generates a 64-bit or 128-bit odd parity key, or a pair of keys, and returns them in encrypted forms. | X | X | X | X | X | X |
| **Key Generate2** | Generates a variable length key or a pair of keys, and returns them in encrypted forms. | | X | X | X | X | X |
| **Key Import** | Converts any key from importable form into operational form. | X | X | X | X | X | X |

| Table 10. Services that require a CCA coprocessor (continued) | | | | | | | |
|---|---|:-:|:-:|:-:|:-:|:-:|:-:|
| **Service Name** | **Function** | **A** | **B** | **C** | **D** | **E** | **F** |
| **Key Part Import** | Combines the clear key parts of an AKEK and returns the combined key value in an internal key token or an update to the CKDS. | X | X | X | X | X | X |
| **Key Part Import2** | Combines the clear key parts of any key type and returns the combined key value in an internal key token or an update to the CKDS. | | X | X | X | X | X |
| **Key Test**<br>**Key Test Extended** | Generates or verifies a secure verification pattern for keys. CSNBKYT requires the tested key to be in the clear or encrypted under the master key. CSNBKYTX also allows the tested key to be encrypted under a key-encrypting key. | X | X | X | X | X | X |
| **Key Test2** | Generates or verifies a secure verification pattern for keys in the clear, encrypted under the master key, or encrypted under a key-encrypting key. | | X | X | X | X | X |
| **Key Translate** | Uses one key-encrypting key to decipher an input key and then enciphers this using another key-encrypting key. | X | X | X | X | X | X |
| **Key Translate2** | Uses one key-encrypting key to decipher an input key and then enciphers this key by using another key-encrypting key within the secure environment. | | X | X | X | X | X |
| **MAC Generation** | Generates a 4-, 6-, or 8-byte message authentication code (MAC) for a text string that the application program supplies. The MAC can be computed by using either the ANSI X9.9-1 algorithm, the ANSI X9.19 optional double-MAC algorithm, or the EMV padding rules. | X | X | X | X | X | X |
| **MAC Generate2** | Generates a keyed hash message authentication code (HMAC) or a ciphered message authentication code (CMAC) for the message string that is provided as input. | | X | X | X | X | X |
| **MAC Verification** | Verifies a 4-byte, 6-byte, or 8-byte message authentication code (MAC) for a text string that the application program supplies. The MAC is computed by using either the ANSI X9.9-1 algorithm, the ANSI X 9.19 optional double-MAC algorithm, or the EMV padding rules and is compared with a user-supplied MAC. | X | X | X | X | X | X |
| **MAC Verify2** | Verifies a keyed hash message authentication code (HMAC) or a ciphered message authentication code (CMAC) for the message text that is provided as input. | | X | X | X | X | X |

| Table 10. Services that require a CCA coprocessor (continued) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Service Name** | **Function** | **A** | **B** | **C** | **D** | **E** | **F** |
| **MDC Generation** | Generates a 128-bit modification detection code (MDC) for a text string that the application program supplies. | X | X | X | X | X | X |
| **Multiple Clear Key Import** | Imports a clear DATA key of one, two, or three parts, enciphers it under the master key, and places the result into an internal key token. | X | X | X | X | X | X |
| **Multiple Secure Key Import** | Enciphers a clear key under the master key or an IMPORTER KEK, and places the result into an internal or external key token as any key type. Permits the import of double-length DATA, MAC, and MACVER keys and triple-length DATA keys. | X | X | X | X | X | X |
| **PCI Interface** | Trusted Key Entry (TKE) workstation interface to the CCA and EP11 coprocessors. | X | X | X | X | X | X |
| **PIN Change/Unblock** | Supports PIN change algorithms that are specified in the VISA Integrated Circuit Card Specifications. | X | X | X | X | X | X |
| **PKA Decrypt** | Decrypts an RSA-encrypted key value and returns it to the application in the clear. | X | X | X | X | X | X |
| **PKA Encrypt** | Encrypts a PKCS 1.2 or ZERO-PAD formatted clear key value under an RSA public key to support Secure Sockets Layer (SSL) applications. | X | X | X | X | X | X |
| **PKA Key Generate** | Generates RSA and ECC keys. | X | X | X | X | X | X |
| **PKA Key Import** | Imports a PKA key token. | X | X | X | X | X | X |
| **PKA Key Token Change** | Changes PKA key tokens (RSA, DSS, and ECC) or trusted block key tokens from encipherment under the cryptographic coprocessor's old RSA master key or ECC master key to encipherment under the current cryptographic coprocessor's RSA master key or ECC master key. | X | X | X | X | X | X |
| **PKA Key Translate** | Translates a source CCA RSA key token into a target external smart card key token. | X | X | X | X | X | X |
| **Prohibit Export** | Modifies an operational key so that it cannot be exported. | X | X | X | X | X | X |
| **Prohibit Export Extended** | Changes the external token of a key in exportable form so that it can be imported at the receiver node, but not exported from that node. | X | X | X | X | X | X |
| **Public Infrastructure Certificate** | Generates a certificate signing request (CSR). | | | | | X | X |

| Table 10. Services that require a CCA coprocessor (continued) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Service Name** | **Function** | **A** | **B** | **C** | **D** | **E** | **F** |
| **Recover PIN From Offset** | Calculates an encrypted customer-entered PIN from a PIN generating key, account information, and an offset, returning the PIN properly formatted and encrypted under a PIN encryption key. | | | X | X | X | X |
| **Remote Key Export** | Generates DES keys for local use and for distribution to an ATM or other remote device. | X | X | X | X | X | X |
| **Restrict Key Attribute** | Modifies an operational key so that it cannot be exported. | | X | X | X | X | X |
| **Retained Key Delete** | Deletes a key that has been retained within a CCA coprocessors. | X | X | X | X | X | X |
| **Retained Key List** | Lists the key labels of keys that have been retained within the CCA coprocessors. | X | X | X | X | X | X |
| **Secure Key Import** | Enciphers a clear key under the master key or an IMPORTER KEK, and places the result into an internal or external key token as any key type. | X | X | X | X | X | X |
| **Secure Key Import2** | Enciphers a variable-length clear HMAC or AES key under the master key and places the result into an internal key token. | | X | X | X | X | X |
| **Secure Messaging for Keys** | Encrypts a text block, including a clear key value decrypted from an internal or external DES token. | X | X | X | X | X | X |
| **Secure Messaging for PINs** | Encrypts a text block, including a clear PIN block recovered from an encrypted PIN block. | X | X | X | X | X | X |
| **SET Block Decompose** | Composes the RSA-OAEP block and the DES-encrypted data block in support of the SET protocol. | X | X | X | X | X | X |
| **SET Block Compose** | Decomposes the RSA-OAEP block and the DES-encrypted data block in support of the SET protocol. | X | X | X | X | X | X |
| **Symmetric Algorithm Decipher** | Deciphers data with the AES algorithm in an address space or a data space using the cipher block chaining or electronic code book modes. | X | X | X | X | X | X |
| **Symmetric Algorithm Encipher** | Enciphers data with the AES algorithm in an address space or a data space using the cipher block chaining or electronic code book modes. | X | X | X | X | X | X |
| **Symmetric Key Export with Data** | Exports a symmetric key encrypted using an RSA key, which is inserted in a PKCS#1 block type 2, with some extra data supplied by the application. | | | X | X | X | X |

| Table 10. Services that require a CCA coprocessor (continued) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Service Name** | **Function** | **A** | **B** | **C** | **D** | **E** | **F** |
| **Symmetric Key Generate** | Generates a symmetric (DATA) key and returns it in two forms: encrypted under the DES master key and encrypted under a PKA public key. | X | X | X | X | X | X |
| **Symmetric Key Import** | Imports a symmetric (DATA) key that is enciphered under an RSA public key and enciphers it under the DES master key. | X | X | X | X | X | X |
| **Symmetric Key Import2** | Imports an HMAC or AES key that is enciphered under an RSA public key or AES EXPORTER key and returns the key in operational form, enciphered under the master key. | | X | X | X | X | X |
| **Symmetric Key Export** | Transfers an application-supplied symmetric key from encryption under the host master key to encryption under an application-supplied RSA public key or AES EXPORTER key. The application-supplied key must be an internal key token or the label in the CKDS of a DES DATA, AES DATA, or variable-length symmetric key token. | X | X | X | X | X | X |
| **Transaction Validation** | Supports the generation and validation of American Express card security codes. | X | X | X | X | X | X |
| **Trusted Block Create** | Creates a trusted block under dual control that is in external form, encrypted under an IMP-PKA transport key. | X | X | X | X | X | X |
| **TR-31 Export** | Converts a CCA token to TR-31 format for export to another party. | | X | X | X | X | X |
| **TR-31 Import** | Converts a TR-31 key block to a CCA token. | | X | X | X | X | X |
| **TR-34 Bind-Begin** | Used for operations that take place at the Key Distribution Host (KDH) during TR-34 Protocol Bind related operations. | | | | | X | X |
| **TR-34 Bind-Complete** | Used for operations that take place at the Key Receiving Device (KRD) during TR-34 Protocol Bind related operations. | | | | | X | X |
| **TR-34 Key Distribution** | Used for operations that take place at the Key Distribution Host (KDH) during TR-34 Protocol Key Transport related operations. | | | | | X | X |
| **TR-34 Key Receive** | Used for operations that take place at the Key Receiving Device (KRD) during TR-34 Protocol Key Transport related operations. | | | | | X | X |

| Table 10. Services that require a CCA coprocessor (continued) | | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|
| **Service Name** | **Function** | **A** | **B** | **C** | **D** | **E** | **F** |
| **Unique Key Derive** | Derives the following key types:<br><br>• CIPHER<br>• ENCIPHER<br>• DECIPHER<br>• MAC<br>• MACVER<br>• IPINENC<br>• OPINENC<br>• DATA token containing a PIN Key | | | X | X | X | X |
| **VISA CVV Generate** | Generates a Card Verification Value (CVV) or Card Verification Code (CVC). | X | X | X | X | X | X |
| **VISA CVV Verify** | Verifies a Card Verification Value (CVV) or Card Verification Code (CVC). | X | X | X | X | X | X |

## PKCS #11 services

PKCS #11 services are services in support of the PKCS #11 API. These services provide some of the functions for the PKCS #11 API and can be called directly.

The ICSF implementation of the PKCS #11 API does not require cryptographic hardware for clear key cryptography. The CP Assist for Cryptographic Functions and CCA coprocessors are used, if available, but are not required.

For secure key cryptography, an active PKCS #11 coprocessor is required. PKCS #11 coprocessors are supported on zEnterprise EC12 and BC12 and later systems with CP Assist for Cryptographic Functions DES/TDES Enablement.

For regional cryptography support, an active Regional Cryptographic Server is required. Regional Cryptographic Server support is available on IBM zEnterprise EC12 or later systems with CP Assist for Cryptographic Functions DES/TDES Enablement.

Dilithium key cryptography support is available on IBM z15 or later.

The following PKCS #11 services use a cryptographic accelerator, if available. Accelerators perform clear, RSA, and Diffie Hellman (DH) operations.

• PKCS #11 Derive key using DH
• PKCS #11 Private Key Sign
• PKCS #11 Public Key Verify
• PKCS #11 Unwrap Key

contains a summary of the PKCS #11 callable services.

| Table 11. Summary of PKCS #11 callable services support | |
|---|---|
| **Service name** | **Function** |
| PKCS #11 Derive Multiple Keys | Generates multiple secret key objects and protocol-dependent keying material from an existing secret key object. |
| PKCS #11 Derive key | Generates a new secret key object from an existing key object. |

*Table 11. Summary of PKCS #11 callable services support (continued)*

| Service name | Function |
|---|---|
| PKCS #11 Get Attribute Value | Lists the attributes of an object. |
| PKCS #11 Generate Key Pair | Generates an RSA, DSA, Elliptic Curve, Diffie-Hellman, or Dilithium key pair. |
| PKCS #11 Generate Secret Key | Generates a secret key or set of domain parameters. |
| PKCS #11 Generate HMAC | Generates a hashed message authentication code (MAC). |
| PKCS #11 Verify HMAC | Verifies a hash message authentication code (MAC). |
| PKCS #11 One-way hash, sign, or verify | Generates a one-way hash on specified text, sign specified text, or verify a signature on specified text. |
| PKCS #11 Private Key Sign | Decrypts or signs data by using an RSA private key that uses zero-pad or PKCS #1 1.5 and 2.1 formatting, signs data by using a DSA private key, signs data by using an Elliptic Curve private key in combination with DSA, or signs data by using a Dilithium private key. |
| PKCS #11 Public Key Verify | Encrypts or verifies data by using an RSA public key that uses zero-pad or PKCS #1 1.5 and 2.1 formatting, verifies a signature by using a DSA public key, verifies a signature by using an Elliptic Curve public key in combination with DSA, or verifies a signature by using a Dilithium public key. |
| PKCS #11 Pseudo-random Function | Generates pseudo-random output of arbitrary length. |
| PKCS #11 Secret Key Decrypt | Deciphers data by using a symmetric key. |
| PKCS #11 Secret Key Encrypt | Enciphers data by using a symmetric key. |
| PKCS #11 Set Attribute Value | Updates the attributes of an object. |
| PKCS #11 Token Record Create | Initializes or reinitializes a z/OS PKCS #11 token, creates or copies a token object in the token data set, or creates or copies a session object for the current PKCS #11 session. |
| PKCS #11 Token Record Delete | Deletes a z/OS PKCS #11 token, token object or session object. |
| PKCS #11 Token Record List | Obtains a list of z/OS PKCS #11 tokens or a list of token and session objects for a token. |
| PKCS #11 Unwrap Key | Unwraps and creates a key object by using another key. |
| PKCS #11 Wrap Key | Wraps a key with another key. |

contains a summary of the services that do not use cryptography and have no hardware requirement.

| Table 12. Services for TKDS | |
|---|---|
| **Service name** | **Function** |
| Key Data Set List (CSFKDSL) | Generates a list of TKDS objects that match a search criteria. |
| Key Data Set Metadata Read (CSFKDMR) | Reads the metadata of a TKDS record. |
| Key Data Set Metadata Write (CSFKDMW) | Changes the metadata of a set of TKDS records. The in-store copy and the DASD copy are updated. |

# Appendix C. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the Contact the z/OS team web page (www.ibm.com/systems/campaignmail/z/zos/contact_z) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

## Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

## Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

## Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Vol I*

## Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1)

are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 \* FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* \* FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

**? indicates an optional syntax element**
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

**! indicates a default syntax element**
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

**\* indicates an optional syntax element that is repeatable**
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

**Notes:**

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.

2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.

3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

**+ indicates a syntax element that must be included**

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

# Notices

This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing*
*IBM Corporation*
*North Castle Drive, MD-NC119*
*Armonk, NY 10504-1785*
*United States of America*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing*
*Legal and Intellectual Property Law*
*IBM Japan Ltd.*
*19-21, Nihonbashi-Hakozakicho, Chuo-ku*
*Tokyo 103-8510, Japan*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for the Knowledge Centers. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Corporation*
*Site Counsel*
*2455 South Road*

*Poughkeepsie, NY 12601-5400*
*USA*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

# Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

## Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

## Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

## Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

**Rights**

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

# IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

# Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, JES3, and MVS™, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

# Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those

products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

## Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and Trademark information (www.ibm.com/legal/copytrade.shtml).

# Glossary

This glossary defines terms and abbreviations used in Integrated Cryptographic Service Facility (ICSF).

This glossary includes terms and definitions from:

- The American National Standard Dictionary for Information Technology, ANSI INCITS 172, by the American National Standards Institute (ANSI). Copies can be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.
- The Information Technology Vocabulary, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

Definitions specific to the Integrated Cryptographic Services Facility are labeled "In ICSF."

**access method services (AMS)**
The facility used to define and reproduce VSAM key-sequenced data sets (KSDS).

**Advanced Encryption Standard (AES)**
In computer security, the National Institute of Standards and Technology (NIST) Advanced Encryption Standard (AES) algorithm.

**AES**
Advanced Encryption Standard.

**American National Standard Code for Information Interchange (ASCII)**
The standard code using a coded character set consisting of 7-bit characters (8 bits including parity check) that is used for information exchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

**ANSI X9.19**
An ANSI standard that specifies an optional double-MAC procedure which requires a double-length MAC key.

**application program**
A program written for or by a user that applies to the user's work, such as a program that does inventory control or payroll.

A program used to connect and communicate with stations in a network, enabling users to perform application-oriented activities.

**application program interface (API)**
A functional interface supplied by the operating system or by a separately orderable licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program.

In ICSF, a callable service.

**asymmetric cryptography**
Synonym for public key cryptography.

**authentication pattern**
An 8-byte pattern that ICSF calculates from the master key when initializing the cryptographic key data set. ICSF places the value of the authentication pattern in the header record of the cryptographic key data set.

**authorized program facility (APF)**
A facility that permits identification of programs authorized to use restricted functions.

**callable service**

A predefined sequence of instructions invoked from an application program, using a CALL instruction. In ICSF, callable services perform cryptographic functions and utilities.

**CBC**

Cipher block chaining.

**CCA**

Common Cryptographic Architecture.

**CCF**

Cryptographic Coprocessor Feature.

**CDMF**

Commercial Data Masking Facility.

**CEDA**

A CICS transaction that defines resources online. Using CEDA, you can update both the CICS system definition data set (CSD) and the running CICS system.

**Central Credit Committee**

The official English name for *Zentraler Kreditausschuss*, also known as ZKA. ZKA was founded in 1932 and was renamed in August 2011 to *Die Deutsche Kreditwirtschaft*, also known as DK. DK is an association of the German banking industry. The hybrid term in English for DK is 'German Banking Industry Committee'.

**CEX2A**

Crypto Express2 Accelerator

**CEX2C**

Crypto Express2 Coprocessor

**CEX3A**

Crypto Express3 Accelerator

**CEX3C**

Crypto Express3 Coprocessor

**CEX4A**

Crypto Express4 Accelerator

**CEX4C**

Crypto Express4 CCA Coprocessor

**CEX4P**

Crypto Express4 PKCS #11 Coprocessor

**CEX5A**

Crypto Express5 Accelerator

**CEX5C**

Crypto Express5 CCA Coprocessor

**CEX5P**

Crypto Express5 PKCS #11 Coprocessor

**CEX6A**

Crypto Express6 Accelerator

**CEX6C**

Crypto Express6 CCA Coprocessor

**CEX6P**

Crypto Express6 PKCS #11 Coprocessor

**CEX7A**

Crypto Express7 Accelerator

**CEX7C**

Crypto Express7 CCA Coprocessor

**CEX7P**
Crypto Express7 PKCS #11 Coprocessor

**checksum**
The sum of a group of data associated with the group and used for checking purposes. (T)

In ICSF, the data used is a key part. The resulting checksum is a two-digit value you enter when you enter a master key part.

**Chinese Remainder Theorem (CRT)**
A mathematical theorem that defines a format for the RSA private key that improves performance.

**CICS**
Customer Information Control System.

**cipher block chaining (CBC)**
A mode of encryption that uses the data encryption algorithm and requires an initial chaining vector. For encipher, it exclusively ORs the initial block of data with the initial control vector and then enciphers it. This process results in the encryption both of the input block and of the initial control vector that it uses on the next input block as the process repeats. A comparable chaining process works for decipher.

**ciphertext**
In computer security, text produced by encryption.

Synonym for enciphered data.

**CKDS**
Cryptographic Key Data Set.

**clear key**
Any type of encryption key not protected by encryption under another key.

**CMOS**
Complementary metal oxide semiconductor.

**coexistence mode**
An ICSF method of operation during which CUSP or PCF can run independently and simultaneously on the same ICSF system. A CUSP or PCF application program can run on ICSF in this mode if the application program has been reassembled.

**Commercial Data Masking Facility (CDMF)**
A data-masking algorithm using a DES-based kernel and a key that is shortened to an effective key length of 40 DES key-bits. Because CDMF is not as strong as DES, it is called a masking algorithm rather than an encryption algorithm. Implementations of CDMF, when used for data confidentiality, are generally exportable from the USA and Canada.

**Common Cryptographic Architecture: Cryptographic Application Programming Interface**
Defines a set of cryptographic functions, external interfaces, and a set of key management rules that provide a consistent, end-to-end cryptographic architecture across different IBM platforms.

**compatibility mode**
An ICSF method of operation during which a CUSP or PCF application program can run on ICSF without recompiling it. In this mode, ICSF cannot run simultaneously with CUSP or PCF.

**complementary keys**
A pair of keys that have the same clear key value, are different but complementary types, and usually exist on different systems.

**console**
A part of a computer used for communication between the operator or maintenance engineer and the computer. (A)

**control-area split**
In systems with VSAM, the movement of the contents of some of the control intervals in a control area to a newly created control area in order to facilitate insertion or lengthening of a data record when there are no remaining free control intervals in the original control area.

**control block**
A storage area used by a computer program to hold control information. (I) Synonymous with control area.

The circuitry that performs the control functions such as decoding microinstructions and generating the internal control signals that perform the operations requested. (A)

**control interval**
A fixed-length area of direct-access storage in which VSAM stores records and creates distributed free space. Also, in a key-sequenced data set or file, the set of records pointed to by an entry in the sequence-set index record. The control interval is the unit of information that VSAM transmits to or from direct access storage. A control interval always comprises an integral number of physical records.

**control interval split**
In systems with VSAM, the movement of some of the stored records in a control interval to a free control interval to facilitate insertion or lengthening of a record that does not fit in the original control interval.

**control statement input data set**
A key generator utility program data set containing control statements that a particular key generator utility program job will process.

**control statement output data set**
A key generator utility program data set containing control statements to create the complements of keys created by the key generator utility program.

**control vector**
In ICSF, a mask that is exclusive ORed with a master key or a transport key before ICSF uses that key to encrypt another key. Control vectors ensure that keys used on the system and keys distributed to other systems are used for only the cryptographic functions for which they were intended.

**CPACF**
CP Assist for Cryptographic Functions

**CP Assist for Cryptographic Functions**
Implemented on all IBM servers to provide AES and DES encryption and SHA-1 secure hashing.

**cross memory mode**
Synchronous communication between programs in different address spaces that permits a program residing in one address space to access the same or other address spaces. This synchronous transfer of control is accomplished by a calling linkage and a return linkage.

**CRT**
Chinese Remainder Theorem.

**Crypto Express2 Coprocessor**
An asynchronous cryptographic coprocessor available on the z9 EC, z9 BC, z10 EC and z10 BC.

**Crypto Express3 Coprocessor**
An asynchronous cryptographic coprocessor available on z10 EC, z10 BC, z114, z196, zEC12, and zBC12.

**Crypto Express4 Coprocessor**
An asynchronous cryptographic coprocessor available on zEC12 and zBC12.

**Crypto Express5 Coprocessor**
An asynchronous cryptographic coprocessor available on z13 and z13s.

**Crypto Express6 Coprocessor**
An asynchronous cryptographic coprocessor available on z14 and z14 ZR1.

**Crypto Express7 Coprocessor**
An asynchronous cryptographic coprocessor available on z15.

**cryptographic adapter (4764, 4765, and 4767)**
An expansion board that provides a comprehensive set of cryptographic functions for the network security processor and the workstation in the TSS family of products.

**cryptographic coprocessor**
A tamper responding, programmable, cryptographic PCI card, containing CPU, encryption hardware, RAM, persistent memory, hardware random number generator, time of day clock, infrastructure firmware, and software.

**cryptographic key data set (CKDS)**
A data set that contains the encrypting keys used by an installation.

In ICSF, a VSAM data set that contains all the cryptographic keys. Besides the encrypted key value, an entry in the cryptographic key data set contains information about the key.

**cryptography**
The transformation of data to conceal its meaning.

In computer security, the principles, means, and methods for encrypting plaintext and decrypting ciphertext.

In ICSF, the use of cryptography is extended to include the generation and verification of MACs, the generation of MDCs and other one-way hashes, the generation and verification of PINs, and the generation and verification of digital signatures.

**CUSP (Cryptographic Unit Support Program)**
The IBM cryptographic offering, program product 5740-XY6, using the channel-attached 3848. CUSP is no longer in service.

**CUSP/PCF conversion program**
A program, for use during migration from CUSP or PCF to ICSF, that converts a CUSP or PCF cryptographic key data set into a ICSF cryptographic key data set.

**Customer Information Control System (CICS)**
An IBM licensed program that enables transactions entered at remote terminals to be processed concurrently by user written application programs. It includes facilities for building, using, and maintaining databases.

**CVC**
Card verification code used by MasterCard.

**CVV**
Card verification value used by VISA.

**data encryption algorithm (DEA)**
In computer security, a 64-bit block cipher that uses a 64-bit key, of which 56 bits are used to control the cryptographic process and 8 bits are used for parity checking to ensure that the key is transmitted properly.

**data encryption standard (DES)**
In computer security, the National Institute of Standards and Technology (NIST) Data Encryption Standard, adopted by the U.S. government as Federal Information Processing Standard (FIPS) Publication 46, which allows only hardware implementations of the data encryption algorithm.

**data key or data-encrypting key**
A key used to encipher, decipher, or authenticate data.

In ICSF, a 64-bit encryption key used to protect data privacy using the DES algorithm. AES data keys are now supported by ICSF.

**data set**
The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

**data-translation key**
A 64-bit key that protects data transmitted through intermediate systems when the originator and receiver do not share the same key.

**DEA**
Data encryption algorithm.

**decipher**
To convert enciphered data in order to restore the original data. (T)

In computer security, to convert ciphertext into plaintext by means of a cipher system.

To convert enciphered data into clear data. Contrast with encipher. Synonymous with decrypt.

**decode**

To convert data by reversing the effect of some previous encoding. (I) (A)

In ICSF, to decipher data by use of a clear key.

**decrypt**

See decipher.

**DES**

Data Encryption Standard.

**diagnostics data set**

A key generator utility program data set containing a copy of each input control statement followed by a diagnostic message generated for each control statement.

**digital signature**

In public key cryptography, information created by using a private key and verified by using a public key. A digital signature provides data integrity and source nonrepudiation.

**Digital Signature Algorithm (DSA)**

A public key algorithm for digital signature generation and verification used with the Digital Signature Standard.

**Digital Signature Standard (DSS)**

A standard describing the use of algorithms for digital signature purposes. One of the algorithms specified is DSA (Digital Signature Algorithm).

**Dilithium**

A quantum-safe cryptographic algorithm. Also known as LI2.

**DK**

*Die Deutsche Kreditwirtschaft* (German Banking Industry Committee). Formerly known as ZKA.

**domain**

That part of a network in which the data processing resources are under common control. (T)

In ICSF, an index into a set of master key registers.

**DSA**

Digital Signature Algorithm.

**DSS**

Digital Signature Standard.

**ECB**

Electronic codebook.

**ECC**

Elliptic Curve Cryptography.

**ECI**

Eurocheque International S.C., a financial institution consortium that has defined three PIN block formats.

**EID**

Environment Identification.

**electronic codebook (ECB) operation**

A mode of operation used with block cipher cryptographic algorithms in which plaintext or ciphertext is placed in the input to the algorithm and the result is contained in the output of the algorithm.

A mode of encryption using the data encryption algorithm, in which each block of data is enciphered or deciphered without an initial chaining vector. It is used for key management functions and the encode and decode callable services.

**electronic funds transfer system (EFTS)**

A computerized payment and withdrawal system used to transfer funds from one account to another and to obtain related financial data.

**encipher**
To scramble data or to convert data to a secret code that masks the meaning of the data to any unauthorized recipient. Synonymous with encrypt.

Contrast with decipher.

**enciphered data**
Data whose meaning is concealed from unauthorized users or observers.

**encode**
To convert data by the use of a code in such a manner that reconversion to the original form is possible. (T)

In computer security, to convert plaintext into an unintelligible form by means of a code system.

In ICSF, to encipher data by use of a clear key.

**encrypt**
See encipher.

**exit**
To execute an instruction within a portion of a computer program in order to terminate the execution of that portion. Such portions of computer programs include loops, subroutines, modules, and so on. (T)

In ICSF, a user-written routine that receives control from the system during a certain point in processing—for example, after an operator issues the START command.

**exportable form**
A condition a key is in when enciphered under an exporter key-encrypting key. In this form, a key can be sent outside the system to another system. A key in exportable form cannot be used in a cryptographic function.

**exporter key-encrypting key**
A 128-bit key used to protect keys sent to another system. A type of transport key.

**file**
A named set of records stored or processed as a unit. (T)

**GBP**
German Bank Pool.

**German Bank Pool (GBP)**
A German financial institution consortium that defines specific methods of PIN calculation.

**German Banking Industry Committee**
A hybrid term in English for *Die Deutsche Kreditwirtschaft*, also known as DK, an association of the German banking industry. Prior to August 2011, DK was named ZKA for *Zentraler Kreditausschuss*, or Central Credit Committee. ZKA was founded in 1932.

**hashing**
An operation that uses a one-way (irreversible) function on data, usually to reduce the length of the data and to provide a verifiable authentication value (checksum) for the hashed data.

**header record**
A record containing common, constant, or identifying information for a group of records that follows.

**ICSF**
Integrated Cryptographic Service Facility.

**importable form**
A condition a key is in when it is enciphered under an importer key-encrypting key. A key is received from another system in this form. A key in importable form cannot be used in a cryptographic function.

**importer key-encrypting key**
A 128-bit key used to protect keys received from another system. A type of transport key.

**initial chaining vector (ICV)**
A 64-bit random or pseudo-random value used in the cipher block chaining mode of encryption with the data encryption algorithm.

**initial program load (IPL)**
　　The initialization procedure that causes an operating system to commence operation.

　　The process by which a configuration image is loaded into storage at the beginning of a work day or after a system malfunction.

　　The process of loading system programs and preparing a system to run jobs.

**input PIN-encrypting key**
　　A 128-bit key used to protect a PIN block sent to another system or to translate a PIN block from one format to another.

**installation exit**
　　See exit.

**Integrated Cryptographic Service Facility (ICSF)**
　　A licensed program that runs under MVS/System Product 3.1.3, or higher, or OS/390 Release 1, or higher, or z/OS, and provides access to the hardware cryptographic feature for programming applications. The combination of the hardware cryptographic feature and ICSF provides secure high-speed cryptographic services.

**International Organization for Standardization**
　　An organization of national standards bodies from many countries, established to promote the development of standards to facilitate the international exchange of goods and services and to develop cooperation in intellectual, scientific, technological, and economic activity. ISO has defined certain standards relating to cryptography and has defined two PIN block formats.

**ISO**
　　International Organization for Standardization.

**job control language (JCL)**
　　A control language used to identify a job to an operating system and to describe the job's requirements.

**key-encrypting key (KEK)**
　　In computer security, a key used for encryption and decryption of other keys.

　　In ICSF, a master key or transport key.

**key generator utility program (KGUP)**
　　A program that processes control statements for generating and maintaining keys in the cryptographic key data set.

**key output data set**
　　A key generator utility program data set containing information about each key that the key generator utility program generates except an importer key for file encryption.

**key part**
　　A 32-digit hexadecimal value that you enter for ICSF to combine with other values to create a master key or clear key.

**key part register**
　　A register in a cryptographic coprocessor that accumulates key parts as they are entered via TKE.

**key store policy**
　　Ensures that only authorized users and jobs can access secure key tokens that are stored in one of the ICSF key stores - the CKDS or the PKDS.

**key store policy controls**
　　Resources that are defined in the XFACILIT class. A control can verify the caller has authority to use a secure token and identify the action to take when the secure token is not stored in the CKDS or PKDS.

**LI2**
　　Abbreviation for the Dilithium quantum-safe algorithm.

**linkage**
　　The coding that passes control and parameters between two routines.

**load module**
All or part of a computer program in a form suitable for loading into main storage for execution. A load module is usually the output of a linkage editor. (T)

**LPAR mode**
The central processor mode that enables the operator to allocate the hardware resources among several logical partitions.

**MAC generation key**
A 64-bit or 128-bit key used by a message originator to generate a message authentication code sent with the message to the message receiver.

**MAC verification key**
A 64-bit or 128-bit key used by a message receiver to verify a message authentication code received with a message.

**magnetic tape**
A tape with a magnetizable layer on which data can be stored. (T)

**master key**
In computer security, the top-level key in a hierarchy of key-encrypting keys.

ICSF uses master keys to encrypt operational keys. Master keys are known only to the cryptographic coprocessors and are maintained in tamper proof cryptographic coprocessors.

**master key concept**
The idea of using a single cryptographic key, the master key, to encrypt all other keys on the system.

**master key register**
A register in the cryptographic coprocessors that stores the master key that is active on the system.

**master key variant**
A key derived from the master key by use of a control vector. It is used to force separation by type of keys on the system.

**MD5**
Message Digest 5. A hash algorithm.

**message authentication code (MAC)**
The cryptographic result of block cipher operations on text or data using the cipher block chain (CBC) mode of operation.

In ICSF, a MAC is used to authenticate the source of the message, and verify that the message was not altered during transmission or storage.

**modification detection code (MDC)**
A 128-bit value that interrelates all bits of a data stream so that the modification of any bit in the data stream results in a new MDC.

In ICSF, an MDC is used to verify that a message or stored data has not been altered.

**multiple encipherment**
The method of encrypting a key under a double-length key-encrypting key.

**new master key register**
A register in a cryptographic coprocessor that stores a master key before you make it active on the system.

**NIST**
U.S. National Institute of Science and Technology.

**NOCV processing**
Process by which the key generator utility program or an application program encrypts a key under a transport key itself rather than a transport key variant.

**noncompatibility mode**
An ICSF method of operation during which CUSP or PCF can run independently and simultaneously on the same z/OS, OS/390, or MVS system. You cannot run a CUSP or PCF application program on ICSF in this mode.

**nonrepudiation**
> A method of ensuring that a message was sent by the appropriate individual.

**OAEP**
> Optimal asymmetric encryption padding.

**offset**
> The process of exclusively ORing a counter to a key.

**old master key register**
> A register in a cryptographic coprocessor that stores a master key that you replaced with a new master key.

**operational form**
> The condition of a key when it is encrypted under the master key so that it is active on the system.

**output PIN-encrypting key**
> A 128-bit key used to protect a PIN block received from another system or to translate a PIN block from one format to another.

**PAN**
> Personal Account Number.

**parameter**
> Data passed between programs or procedures.

**parmlib**
> A system parameter library, either SYS1.PARMLIB or an installation-supplied library.

**partitioned data set (PDS)**
> A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

**Personal Account Number (PAN)**
> A Personal Account Number identifies an individual and relates that individual to an account at a financial institution. It consists of an issuer identification number, customer account number, and one check digit.

**personal identification number (PIN)**
> The 4- to 12-digit number entered at an automatic teller machine to identify and validate the requester of an automatic teller machine service. Personal identification numbers are always enciphered at the device where they are entered, and are manipulated in a secure fashion.

**Personal Security card**
> An ISO-standard "smart card" with a microprocessor that enables it to perform a variety of functions such as identifying and verifying users, and determining which functions each user can perform.

**PIN block**
> A 64-bit block of data in a certain PIN block format. A PIN block contains both a PIN and other data.

**PIN generation key**
> A 128-bit key used to generate PINs or PIN offsets algorithmically.

**PIN key**
> A 128-bit key used in cryptographic functions to generate, transform, and verify the personal identification numbers.

**PIN offset**
> For 3624, the difference between a customer-selected PIN and an institution-assigned PIN. For German Bank Pool, the difference between an institution PIN (generated with an institution PIN key) and a pool PIN (generated with a pool PIN key).

**PIN verification key**
> A 128-bit key used to verify PINs algorithmically.

**PKA**
> Public Key Algorithm.

**PKCS**
> Public Key Cryptographic Standards (RSA Data Security, Inc.)

**PKDS**
Public key data set (PKA cryptographic key data set).

**plaintext**
Data in normal, readable form.

**primary space allocation**
An area of direct access storage space initially allocated to a particular data set or file when the data set or file is defined. See also secondary space allocation.

**private key**
In computer security, a key that is known only to the owner and used with a public key algorithm to decrypt data or generate digital signatures. The data is encrypted and the digital signature is verified using the related public key.

**processor complex**
A configuration that consists of all the machines required for operation.

**Processor Resource/Systems Manager**
Enables logical partitioning of the processor complex, may provide additional byte-multiplexer channel capability, and supports the VM/XA System Product enhancement for Multiple Preferred Guests.

**Programmed Cryptographic Facility (PCF)**
An IBM licensed program that provides facilities for enciphering and deciphering data and for creating, maintaining, and managing cryptographic keys.

The IBM cryptographic offering, program product 5740-XY5, using software only for encryption and decryption. This product is no longer in service; ICSF is the replacement product.

**PR/SM**
Processor Resource/Systems Manager.

**public key**
In computer security, a key made available to anyone who wants to encrypt information using the public key algorithm or verify a digital signature generated with the related private key. The encrypted data can be decrypted only by use of the related private key.

**public key algorithm (PKA)**
In computer security, an asymmetric cryptographic process in which a public key is used for encryption and digital signature verification and a private key is used for decryption and digital signature generation.

**public key cryptography**
In computer security, cryptography in which a public key is used for encryption and a private key is used for decryption. Synonymous with asymmetric cryptography.

**QSA**
Quantum-safe algorithm. Examples include CRYSTALS-Dilithium Digital Signature Algorithm.

**RACE Integrity Primitives Evaluatiuon Message Digest**
A hash algorithm.

**RCE redirection**
Ability for an existing CCA service to be used in such a way that a request is reformatted and routed to a Regional Cryptographic Server for processing. The RCE redirection function is available when the system has only an RCE device active or the user has READ access to CSF.RCE.FUNCTION.REDIRECT in the XFACILIT SAF resource class.

**RCS**
Regional Cryptographic Server.

**RDO**
Resource definition online.

**record chaining**
When there are multiple cipher requests and the output chaining vector (OCV) from the previous encipher request is used as the input chaining vector (ICV) for the next encipher request.

**Resource Access Control Facility (RACF)**
An IBM licensed program that provides for access control by identifying and verifying the users to the system, authorizing access to protected resources, logging the detected unauthorized attempts to enter the system, and logging the detected accesses to protected resources.

**retained key**
A private key that is generated and retained within the secure boundary of the Crypto Express adapter.

**return code**
A code used to influence the execution of succeeding instructions. (A)

A value returned to a program to indicate the results of an operation requested by that program.

**Rivest-Shamir-Adleman (RSA) algorithm**
A process for public key cryptography that was developed by R. Rivest, A. Shamir, and L. Adleman.

**RMF**
Resource Manager Interface.

**RMI**
Resource Measurement Facility.

**RSA**
Rivest-Shamir-Adleman.

**RSA-PSS**
RSA-Probabilistic Signature Scheme. RSA-PSS is a signature scheme that is based on the RSA cryptosystem and provides increased security assurance. It was added in version 2.1 of PKCS #1.

**SAF**
System Authorization Facility.

**save area**
Area of main storage in which contents of registers are saved. (A)

**secondary space allocation**
In systems with VSAM, area of direct access storage space allocated after primary space originally allocated is exhausted. See also primary space allocation.

**Secure Electronic Transaction**
A standard created by Visa International and MasterCard for safe-guarding payment card purchases made over open networks.

**secure key**
A key that is encrypted under a master key. When ICSF uses a secure key, it is passed to a cryptographic coprocessor where the coprocessor decrypts the key and performs the function. The secure key never appears in the clear outside of the cryptographic coprocessor.

**Secure Sockets Layer**
A security protocol that provides communications privacy over the Internet by allowing client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

**sequential data set**
A data set whose records are organized on the basis of their successive physical positions, such as on magnetic tape.

**SET**
Secure Electronic Transaction.

**SHA (Secure Hash Algorithm, FIPS 180)**
(Secure Hash Algorithm, FIPS 180) The SHA (Secure Hash Algorithm) family is a set of related cryptographic hash functions designed by the National Security Agency (NSA) and published by the National Institute of Standards and Technology (NIST). The first member of the family, published in 1993, is officially called SHA. However, today, it is often unofficially called SHA-0 to avoid confusion with its successors. Two years later, SHA-1, the first successor to SHA, was published. Four more variants, have since been published with increased output ranges and a slightly different design: SHA-224, SHA-256, SHA-384, and SHA-512 (all are sometimes referred to as SHA-2).

**SHA-1 (Secure Hash Algorithm 1, FIPS 180)**
　　A hash algorithm required for use with the Digital Signature Standard.

**SHA-2 (Secure Hash Algorithm 2, FIPS 180)**
　　Four additional variants to the SHA family, with increased output ranges and a slightly different design: SHA-224, SHA-256, SHA-384, and SHA-512 (all are sometimes referred to as SHA-2).

**SHA-3 (Secure Hash Algorithm 3, FIPS 202)**
　　SHA-3 is a subset of the cryptographic primitive family Keccak and is used to build instances of Permutation-Based Hash and Extendable-Output Functions (see also SHAKE). Because of the successful attacks on MD5, SHA-0, and SHA-1, NIST perceived a need for an alternative, dissimilar cryptographic hash, which became SHA-3.

**SHA-224**
　　One of the SHA-2 algorithms.

**SHA-256**
　　One of the SHA-2 algorithms.

**SHA-384**
　　One of the SHA-2 algorithms.

**SHA-512**
　　One of the SHA-2 algorithms.

**SHA3-224**
　　An instance of the SHA-3 algorithm that provides a Permutation-Based Hash.

**SHA3-256**
　　An instance of the SHA-3 algorithm that provides a Permutation-Based Hash.

**SHA3-384**
　　An instance of the SHA-3 algorithm that provides a Permutation-Based Hash.

**SHA3-512**
　　An instance of the SHA-3 algorithm that provides a Permutation-Based Hash.

**SHAKE (combination of Secure Hash Algorithm and Keccak)**
　　A set of Extendable-Output Functions defined in FIPS PUB 202.

**SHAKE128**
　　An instance of the SHA-3 algorithm that provides an Extendable-Output Function.

**SHAKE256**
　　An instance of the SHA-3 algorithm that provides an Extendable-Output Function.

**smart card**
　　A plastic card that has a microchip capable of storing data or process information.

**special secure mode**
　　An alternative form of security that allows you to enter clear keys with the key generator utility program or generate clear PINs.

**SSL**
　　Secure Sockets Layer.

**supervisor state**
　　A state during which a processing unit can execute input/output and other privileged instructions.

**System Authorization Facility (SAF)**
　　An interface to a system security system like the Resource Access Control Facility (RACF).

**system key**
　　A key that ICSF creates and uses for internal processing.

**System Management Facility (SMF)**
　　A base component of z/OS that provides the means for gathering and recording information that can be used to evaluate system usage.

**TDEA**
　　Triple Data Encryption Algorithm.

**TKE**
Trusted key entry.

**Transaction Security System**
An IBM product offering including both hardware and supporting software that provides access control and basic cryptographic key-management functions in a network environment. In the workstation environment, this includes the 4755 Cryptographic Adapter, the Personal Security Card, the 4754 Security Interface Unit, the Signature Verification feature, the Workstation Security Services Program, and the AIX Security Services Program/6000. In the host environment, this includes the 4753 Network Security Processor and the 4753 Network Security Processor MVS Support Program.

**transport key**
A key used to protect keys distributed from one system to another. A transport key can be an AES or DES key-encrypting key (importer or exporter).

**transport key variant**
A key derived from a transport key by use of a control vector. It is used to force separation by type for keys sent between systems.

**TRUE**
Task-related User Exit (CICS). The CICS-ICSF Attachment Facility provides a CSFATRUE and CSFATREN routine.

**UAT**
UDX Authority Table.

**UDF**
User-defined function.

**UDK**
User-derived key.

**UDP**
User Developed Program.

**UDX**
User Defined Extension.

**verification pattern**
An 8-byte pattern that ICSF calculates from the key parts you enter when you enter a master key or clear key. You can use the verification pattern to verify that you have entered the key parts correctly and specified a certain type of key.

**Virtual Storage Access Method (VSAM)**
An access method for indexed or sequential processing of fixed and variable-length records on direct-access devices. The records in a VSAM data set or file can be organized in logical sequence by means of a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by means of relative-record number.

**Virtual Telecommunications Access Method (VTAM)**
An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

**VISA**
A financial institution consortium that has defined four PIN block formats and a method for PIN verification.

**VISA PIN Verification Value (VISA PVV)**
An input to the VISA PIN verification process that, in practice, works similarly to a PIN offset.

**3621**
A model of an IBM Automatic Teller Machine that has a defined PIN block format.

**3624**
A model of an IBM Automatic Teller Machine that has a defined PIN block format and methods of PIN calculation.

**4764**

    The IBM 4764 PCI-X Cryptographic Coprocessor processor provides a secure programming and hardware environment where AES, DES, and RSA processes are performed.

**4765**

    The IBM 4765 PCIe Cryptographic Coprocessor processor provides a secure programming and hardware environment where AES, DES, ECC, and RSA processes are performed.

**4767**

    The IBM 4767 PCIe Cryptographic Coprocessor processor provides a secure programming and hardware environment where AES, DES, ECC, and RSA processes are performed.

# Index

shortcut keys 79
signature master key (SMK) 29
single-length key
    using 25
SMF records
    generated by ICSF 52
special secure mode
    allows clear keys and PINs 51
    description 31
    installation option 19
    using for clear key entry 31
    z9 BC 31
    z9 EC 31
starting ICSF
    exits 19
stopping ICSF
    exits 19
symmetric cryptographic system 2

## T

TKDS (token data set)
    description 37
TKE workstation 9, 13, 17
token data set (TKDS)
    description 37
trademarks 86
translating ciphertext 23
transport key
    example of use 29
    how used 11
    used to send cryptographic keys to other installations
    51
    variant 41
transporting data across a network 9
triple-length key 29
trusted key entry 9, 13
types of AES keys 26
types of DES keys 25

## U

UDX option 19
updating the CKDS 16, 31
updating the PKDS 16
usage tracking 21
user interface
    ISPF 79
    TSO/E 79
using different configurations 48
using ICSF exits to meet special needs 18
using RSA encryption 30

## V

V2R2 changed information FMID HCR77C0 xix
V2R2 deleted information FMID HCR77C0 xix
V2R2 new information FMID HCR77C0 xix
V2R3 changed information FMID HCR77C1 xix
V2R3 changed information FMID HCR77D0 xviii
V2R3 deleted information FMID HCR77C1 xix
V2R3 deleted information FMID HCR77D0 xviii
V2R3 new information FMID HCR77C1 xviii

V2R3 new information FMID HCR77D0 xviii
V2R4 changed information FMID HCR77D1 xvii
V2R4 deleted information FMID HCR77D1 xviii
V2R4 new information FMID HCR77D1 xvii
verifying
    customer identity 26
    PINs 23
virtual storage constraint relief
    for the caller of ICSF 42
VISA card-verification value (CVV) 6
VISA Format 2 14
VISA Format 3 14
VISA Format 4 14
VISA PIN, through a VISA PIN validation value (VISA PVV) 14
VTAM session-level encryption
    and ICSF 42

## W

WAITLIST option 19

## Z

z/OS ICSF
    operating system requirement 43

**IBM**®

SC14-7505-09