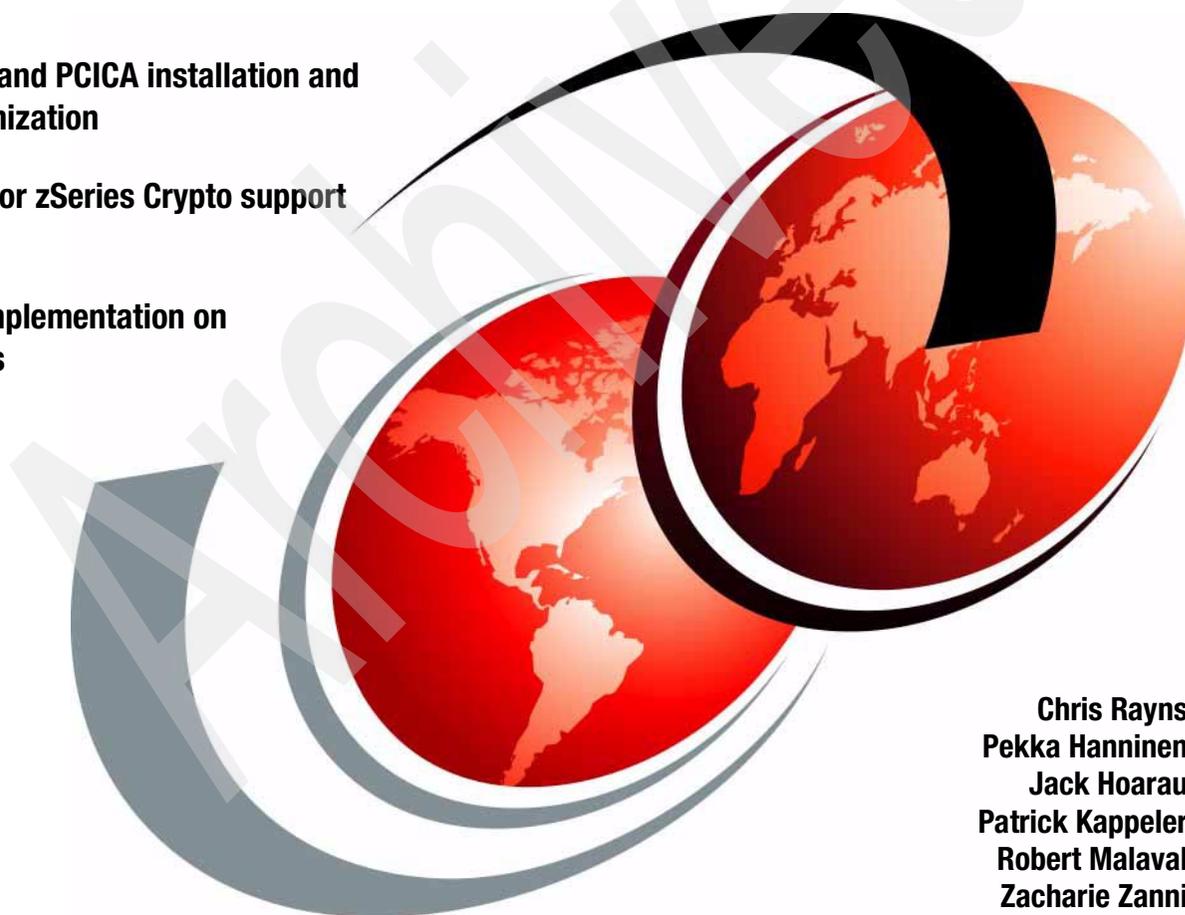


# zSeries Crypto Guide Update

PCICC and PCICA installation and  
customization

Linux for zSeries Crypto support

UDX implementation on  
zSeries



Chris Rayns  
Pekka Hanninen  
Jack Hoarau  
Patrick Kappeler  
Robert Malaval  
Zacharie Zanni





International Technical Support Organization

**zSeries Crypto Guide Update**

April 2003

Archived

**Note:** Before using this information and the product it supports, read the information in “Notices” on page iii.

### **First Edition (March 2003)**

This edition applies to z/OS Version 1, Release 3.

© Copyright International Business Machines Corporation 2003. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Notices

This information was developed for products and services offered in the U.S.A.

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 give you any license to these patents. You can send license inquiries, in writing, to:  
*IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.*

*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.

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

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.

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 illustrates 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. 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 IBM's application programming interfaces.

# Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	Multiprise®	S/390®
CICS®	MVS™	TME®
developerWorks™	OS/2®	VisualAge®
eServer™	OS/390®	VTAM®
@server™	OS/400®	WebSphere®
ES/9000®	PR/SM™	z/OS™
ESCON®	Redbooks™	z/VM™
FICON™	Redbooks (logo)  ™	zSeries™
IBM Payment Gateway™	RACF®	
IBM®	RMF™	

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

# Contents

<b>Notices</b> .....	iii
Trademarks .....	iv
<b>Preface</b> .....	xi
The team that wrote this redbook .....	xi
Become a published author .....	xiii
Comments welcome .....	xiii
<b>Chapter 1. Introduction</b> .....	1
1.1 IBM Cryptographic Common Architecture .....	2
1.2 CCA key management functions .....	2
1.3 Implementing CCA key management concepts in S/390 .....	4
1.3.1 S/390 Cryptographic Coprocessor Facility (CCF) .....	4
1.3.2 S/390 PCI Cryptographic Coprocessor (PCICC) .....	5
1.3.3 S/390 PCI Cryptographic Accelerator (PCICA) .....	5
1.4 S/390 integrated cryptography implementation .....	5
1.4.1 S/390 integrated cryptography implementation .....	7
1.4.2 Enablement of the cryptographic coprocessors .....	8
1.4.3 LPAR domains and Trusted Key Entry (TKE) .....	9
1.5 Crypto support for z/VM™ and Linux .....	11
1.6 Industry standards for cryptographic modules .....	12
<b>Chapter 2. PCICC and PCICA product overview</b> .....	15
2.1 Description of hardware .....	16
2.1.1 Definitions .....	16
2.1.2 Hardware implementation .....	16
2.1.3 Introduction to the S/390 PCI Cryptographic Coprocessors .....	22
2.1.4 PCICC card: physical security, handling, and shipping .....	32
2.2 Adjunct Processor (AP) management .....	37
2.2.1 Introduction to Adjunct Processor architecture .....	37
2.2.2 AP management and PCICC initialization .....	38
2.3 PCICC microcode load .....	40
2.3.1 The IBM 4758 CCA application .....	40
2.3.2 The software hierarchy in the coprocessor .....	41
2.3.3 PCICC microcode patches .....	43
2.3.4 Function Control Vector (FCV) enablement .....	44
2.3.5 Software support of PCICC coprocessors .....	46
2.3.6 The TKE V3.1 Workstation .....	47

<b>Chapter 3. Planning and hardware installation</b> .....	49
3.1 Hardware requirements .....	50
3.1.1 Hardware required by product .....	50
3.2 Feature codes .....	53
3.3 Concurrent PCICC/PCICA installation tasks .....	55
3.3.1 First scenario .....	57
3.3.2 Second scenario (adding PCICC concurrently) .....	83
3.3.3 Third scenario (UDX installation - hardware side) .....	84
3.3.4 Removing one PCICC .....	90
3.4 The z900 channel subsystem .....	91
3.4.1 The z900 internal structure .....	91
3.4.2 View Hardware Configuration icon (CPC configuration task) .....	93
3.5 Planning list items .....	96
3.5.1 Capacity planning considerations .....	96
3.5.2 Installation of the ordered PCICCs .....	96
3.6 PR/SM setup .....	97
3.6.1 Host definitions .....	97
3.6.2 CCF crypto modules, domains, and authority definitions .....	97
3.6.3 Authority signature keys on IBM Personal Security Card (PSC) .....	97
3.6.4 Authority signature key in the TKE Workstation key storage .....	98
3.6.5 IMP-PKA keys in the workstation key storage .....	98
3.6.6 Migration of master or operational key parts on PSC .....	98
3.7 Site security policy .....	98
<b>Chapter 4. Installation, configuration and startup of ICSF</b> .....	101
4.1 PCICC and PCICA card plugging .....	102
4.1.1 PCICC enablement .....	103
4.2 Installing User Defined Extensions (UDX) .....	109
4.3 LPAR setup .....	114
4.3.1 The image profile processor page .....	115
4.3.2 The Crypto page .....	118
4.3.3 The PCI Crypto page .....	121
4.3.4 Changing LPAR Cryptographic controls dynamically .....	123
4.4 Integrated Cryptographic Services Facility (ICSF) setup .....	124
4.4.1 Major changes from previous releases .....	125
4.4.2 Started task and the first time start .....	126
4.4.3 Master Keys .....	127
4.4.4 Initial Master Key entry with the pass phrase initialization utility .....	128
4.4.5 Installation of the PCICC and PCICA cards .....	130
4.4.6 Changing the PKA Master Keys via ICSF panels .....	132
4.4.7 UDX-related definitions in the OPTIONS Data Set .....	141
4.4.8 Installation of the UDX shown in ICSF panels .....	143

<b>Chapter 5. Customizing PCICC and CCF using TKE V3.1</b> . . . . .	147
5.1 Introduction to the TKE V3.1 Workstation . . . . .	148
5.1.1 Major changes . . . . .	148
5.1.2 Before using the new TKE . . . . .	152
5.1.3 The TKE V3.1 software . . . . .	153
5.1.4 TKE Workstation installation - general information . . . . .	153
5.1.5 TKE definitions . . . . .	154
5.2 TKE Workstation TCP/IP setup . . . . .	156
5.2.1 z/OS TCP/IP Host Transaction Program . . . . .	160
5.2.2 TKE Workstation 4758 setup . . . . .	160
5.2.3 TKE access control administration . . . . .	163
5.2.4 Starting the TKE application . . . . .	169
5.3 TKE application: managing host Crypto coprocessors . . . . .	170
5.3.1 Managing modules . . . . .	170
5.3.2 PCICC and CCF setup on the TKE Workstation . . . . .	173
5.3.3 Manage and update the Crypto module notebook on TKE . . . . .	183
5.3.4 PCICC module notebook . . . . .	191
5.3.5 Crypto CCF notebook . . . . .	214
5.3.6 Backing up the TKE files . . . . .	238
5.4 4753 Key Token Migration facility . . . . .	240
<b>Chapter 6. Support functions</b> . . . . .	241
6.1 RACF access control to ICSF services . . . . .	242
6.1.1 New profiles in the CSFSERV class . . . . .	242
6.2 Crypto usage measurement . . . . .	243
6.2.1 SMF record type 82 . . . . .	243
6.2.2 SMF record type 70, subtype 2 . . . . .	249
6.2.3 SMF record type 72, subtype 3 . . . . .	250
6.3 RMF reporting . . . . .	251
<b>Chapter 7. Linux for zSeries support of cryptographic coprocessors</b> . . . . .	255
7.1 Support of hardware coprocessors . . . . .	256
7.1.1 The provided hardware services . . . . .	258
7.2 Access to cryptographic services . . . . .	259
7.2.1 Functions of z90crypt API . . . . .	260
7.2.2 The libica API . . . . .	263
7.2.3 The PKCS#11 API . . . . .	263
7.2.4 Functions of the OpenSSL “engine” . . . . .	264
7.3 Virtualization . . . . .	264
7.3.1 Using a crypto device in VM . . . . .	265
7.4 Our installation with a 31-bit Linux . . . . .	266
7.4.1 Preparation . . . . .	266
7.4.2 Installing the crypto device driver . . . . .	266

7.4.3	Running the modified install script . . . . .	267
7.4.4	Loading the device driver and defining the crypto device node . . . . .	268
7.4.5	Checking the device status . . . . .	269
7.5	Low-level testing . . . . .	270
7.5.1	Installation and test of the Crypto Interface Library (libica) . . . . .	272
7.6	Example SSL-enabled application: Apache Web server . . . . .	277
7.7	Low-level test programs . . . . .	280
7.7.1	testcrtde.c . . . . .	280
7.7.2	icacrtde.c . . . . .	283
7.7.3	tell.h . . . . .	288
7.7.4	tellit.c . . . . .	288
7.7.5	makecrtde . . . . .	293
7.7.6	makeicacr . . . . .	293
	<b>Appendix A. PCICC User Defined Extensions (UDX) . . . . .</b>	<b>295</b>
	UDX overview . . . . .	296
	PCICC code structure and UDX . . . . .	296
	ICSF and PCICC communications . . . . .	298
	UDX invocation . . . . .	298
	UDX function code identifier . . . . .	299
	The UDX callable service and the stub . . . . .	299
	The UDX development process . . . . .	302
	What the UDX does, and how . . . . .	302
	The PCICC UDX development process . . . . .	302
	UDX process phase 2 support . . . . .	303
	PCICC UDX generation process overview . . . . .	303
	Building the UDX coprocessor executable . . . . .	304
	Installing the PCICC UDX . . . . .	305
	Designing and developing the host piece of the UDX . . . . .	305
	The ICSF callable service and the service stub . . . . .	306
	The access control point exit . . . . .	308
	<b>Appendix B. Callable services access control points . . . . .</b>	<b>311</b>
	The access control points . . . . .	311
	Access control points in the PCICC and ICSF . . . . .	316
	New access control points and TKE users . . . . .	316
	Non TKE users . . . . .	316
	New TKE users . . . . .	316
	TKE users . . . . .	316
	<b>Appendix C. Exploitation of the cryptographic coprocessors . . . . .</b>	<b>317</b>
	Exploitation of the zSeries CCFs and PCI coprocessors . . . . .	318
	The IBM exploiters . . . . .	319
	z/OS System SSL . . . . .	320

z/OS Open Cryptographic Services Facility (OCSF) . . . . .	321
IBM HTTP Server for z/OS . . . . .	321
z/OS LDAP server and client . . . . .	322
CICS Transaction Server and CICS Transaction Gateway . . . . .	322
z/OS TN3270 Server . . . . .	322
z/OS Firewall Technologies . . . . .	322
z/OS DCE . . . . .	323
z/OS Network Authentication Service (Kerberos) . . . . .	323
Payment Processing products . . . . .	323
VTAM Session Level Encryption . . . . .	323
RACF . . . . .	324
z/OS Public Key Infrastructure (PKI) Services . . . . .	324
Crypto Based Transactions (CBT) banking solution . . . . .	324
Java cryptography . . . . .	324
<b>Appendix D. Crypto performance considerations</b> . . . . .	<b>325</b>
General considerations for performance of cryptographic operations . . . . .	326
RMF support for Crypto . . . . .	328
<b>Appendix E. TKE host TCP/IP server setup</b> . . . . .	<b>331</b>
The main TCP/IP files to check and modify . . . . .	331
TCPIP.HOSTS.LOCAL . . . . .	331
TCPIP.DATA . . . . .	332
TCPIP.PROFILE . . . . .	332
TKE Host Transaction Program installation . . . . .	334
CSFTTCP started procedure installation . . . . .	334
The CSFTTKE module . . . . .	336
The CSFTHTP3 REXX exec . . . . .	336
Starting the TKE Host Transaction Program . . . . .	337
<b>Related publications</b> . . . . .	<b>339</b>
IBM Redbooks . . . . .	339
Other resources . . . . .	339
Referenced Web sites . . . . .	339
How to get IBM Redbooks . . . . .	340
IBM Redbooks collections . . . . .	340
<b>Index</b> . . . . .	<b>341</b>

Archived

# Preface

This IBM® Redbook is designed to help you understand and implement the z/OS™ Cryptographic PCICC and PCICA cards. Although this book focuses on the enablement of the z/OS PCICC and PCICA products, cryptography and the available services on z/OS are also discussed and explained, with special attention given to the new Trusted Key Entry (TKE 3.1) workstation.

In addition, we look at how Linux for zSeries™ supports the exploitation of the PCICC and PCICA cryptographic coprocessors by using a generic device driver called z90crypt, which routes the cryptographic work to the PCICC or PCICA cards.

We also describe User Defined Extensions (UDX), which is the facility offered by the S/390® and zSeries PCICC, when running under control of ICSF, to enable users to design and implement their own cryptographic services to be executed in the PCICC itself. This provides maximum flexibility to the PCICC user, along with all the protection that the card can offer.

## The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Montpellier Benchmarking Center.

**Chris Rayns** is a Security Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of security.

**Pekka Hanninen** is a Service Specialist in Finland. He has 25 years of experience in IBM Large Systems software. He has worked at IBM for four years. His areas of expertise include RACF®, cryptography and security administration. He holds certificates for CISSP and CISA.

**Jack Hoarau** is an IT Specialist at the New Technology Center; he joined the PSSC in Montpellier in 1999, and is responsible for Linux on zSeries for presale technical support, workshops, wildfire sessions for WebSphere® Application Server enablement on Linux on zSeries, benchmark and POC support, and advance support. He joined IBM in Montpellier in 1970 in manufacturing, then moved to application development in networking and CIM, and participated in field support and international assignments as a systems specialist.

**Patrick Kappeler** is a former computer specialist in the French Air Force and joined IBM in 1970 as a Diagnostic Programs Designer. He has held several specialist and management positions as well as international assignments, all dealing with S/390 technical support. He joined the EMEA S/390 New Technology Center in Montpellier in 1996, where he now provides consulting and presale technical supports in the area of e-business security.

**Robert Malaval** is a Product Engineer at the Montpellier System Laboratory in France. He has 12 years of experience as a Product Engineer in the power area of large systems, and also worked in the CEC area of CMOS for seven years. He holds a degree in Electronics from Montpellier University.

**Zacharie Zanni** is a Product Engineer at the Montpellier Systems Laboratory in France. He has 18 years of experience in Product Engineering, specializing in the CEC area of large systems. He holds a degree in Physics/Chemistry from Strasbourg, France. His areas of expertise include hardware and microcode related to CP, memory, system controller, and crypto chips. He has written extensively on numerous new hardware products. He is currently expanding his area of expertise to system capacity planning and performance.

Thanks also to Andries Mulder of Mulder Training & Consultancy, The Netherlands, and to the following people for their valuable contributions to this project:

Ivan Bailey  
zServer Planning, IBM Poughkeepsie

Manfred Schwengler  
zSeries PSS Microcode Development, IBM Lab Boeblingen

Todd Arnold  
Design of security and cryptographic products, IBM Charlotte

Lucina Green  
ICSF Development, IBM Poughkeepsie

Michael Kelly  
S390 Cryptographic software design, development and service,  
IBM Poughkeepsie

Otto Wohlmuth  
IBM Lab Boeblingen

Virgil Meredith (vendor)  
IBM @server performance analysis, Endicott

Walter H Von Dehsen  
Program manager for zSeries Cryptography & Networking, IBM Poughkeepsie

## Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review redbook form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an Internet note to:

[redbook@us.ibm.com](mailto:redbook@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYJ Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

Archived



# Introduction

In this chapter we provide an overview of IBM cryptography, starting with a discussion of the IBM Common Cryptographic Architecture.

## 1.1 IBM Cryptographic Common Architecture

The IBM Common Cryptographic Architecture, or CCA, defines a set of cryptographic functions, external interfaces, and a set of key management rules which pertain both to the Data Encryption Standard (DES)-based symmetric algorithms and the Public Key Algorithm (PKA) asymmetric algorithms. These provide a consistent, end-to-end, cryptographic architecture across different IBM platforms, such as OS/390®, AIX®, OS/400®, OS/2®, and Windows NT, which conforms to American and International Standards.

Functions of the Common Cryptographic Architecture define services for the following:

- ▶ Key management, which includes generation and exchange of keys securely across networks and between application programs. The exchanged key is securely encrypted using either DES or a Public Key Algorithm used in the context of symmetric key management.
- ▶ Data integrity, with the use of a Message Authentication Code (MAC), Modification Detection Code (MDC), or digital signature.
- ▶ Data confidentiality, with the use of encryption and decryption capabilities accessible at all levels of a network protocol stack.
- ▶ Personal authentication, with PIN generation, verification, and translation.

CCA was introduced in October 1989 with the IBM Transaction Security System and the IBM Integrated Cryptographic Facility (IBM ICSF) with its supporting Integrated Cryptographic Services Facility/MVS™ (ICSF/MVS).

These products and their follow-ons (for ES/9000® ICRF, the CMOS Enterprise Server/Multiprise® Crypto Coprocessor Facility or CCF, the PCICC PCI Cryptographic Coprocessor, and the new PCICA PCI Cryptographic Accelerator available only on zSeries servers and requiring z/OS V1R2) conform to the IBM CCA Application Programming Interface.

## 1.2 CCA key management functions

Key management is essential to successful cryptography. Since the algorithm is usually public knowledge, the security of the data depends on the security of the key used to encipher the data. Enciphered data may be obtained by an adversary, but without access to the cryptographic key, the data remains secure.

Key management in the IBM CCA includes the following:

- ▶ Master Key concept

Each cryptographic system has a Master Key that is kept in the clear inside the *cryptographic facility*, which is a highly secured physical repository. Each operational DES key is encrypted under the appropriate Master Key variant (see next item). This allows an installation to protect many keys while providing physical protection for only one key.

- ▶ PKA keys

The concept of Master Key is also applied to PKA keys that are encrypted under the PKA Master Key.

- ▶ Key separation

Cryptographic keys should be used only for their intended function. For DES keys, the IBM CCA enforces key separation through the use of control vectors (CV).

A *control vector* is a fixed pattern defined for each key type that the cryptographic facility exclusively ORs with the Master Key to produce a Master Key variant that is used to encrypt the key. Effectively, this produces a unique Master Key for each key type. The Master Key variants protect keys operating on the system; these are called *operational keys*.

The control vector concept also applies to the secure transportation of *symmetric* keys, where the transported key is encrypted under a variant of the key-encrypting-key. For example, when a key is stored with a file or sent to another system, the key is encrypted under a key-encrypting key.

Figure 1-1 on page 4 illustrates these concepts, where the data-encryption key and the key-encrypting key are shown as sealed envelopes to indicate that they are encrypted under the corresponding crypto facility Master Key variant, whereas the external key token is encrypted under a key-encrypting-key variant.

**Note:** The control vector concept is not relevant to the PKA keys; instead, “key usage flag bits” are associated to the PKA key to restrict its usage to signature and/or symmetric key management.

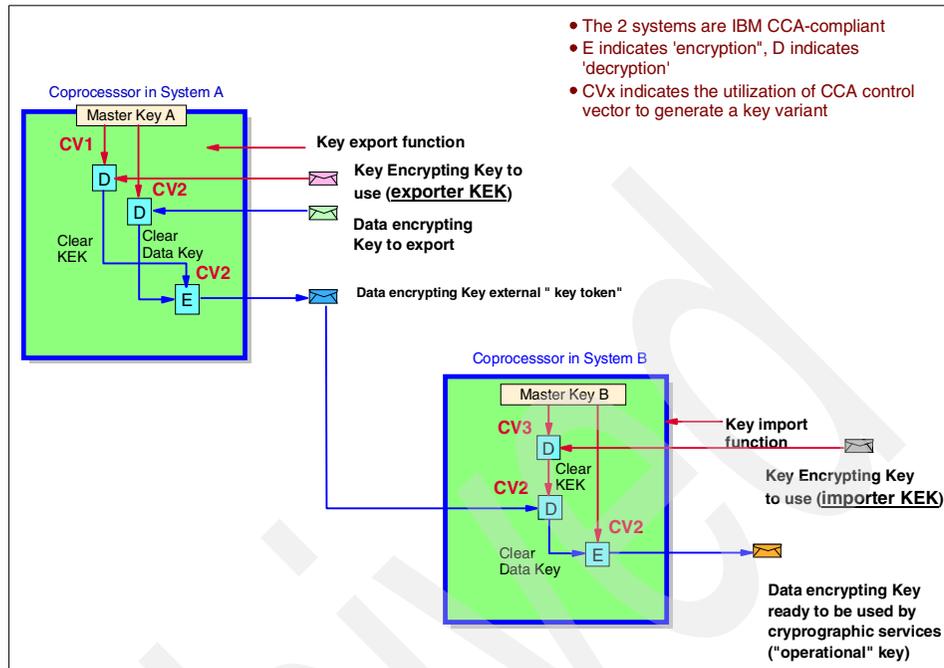


Figure 1-1 IBM CCA Master Key and key separation concepts

Notice the two types of key-encrypting keys: the exporter-KEK on the sending side, and the importer-KEK on the receiving side. These two keys have the same clear value but they have an associated control vector which restricts their function to exportation or importation of other keys.

## 1.3 Implementing CCA key management concepts in S/390

This section describes the concepts underlining CCA key management in S/390.

### 1.3.1 S/390 Cryptographic Coprocessor Facility (CCF)

The S/390 CCF implements three Master Keys:

- ▶ The DES Master Key
- ▶ The PKA Signature Master Key (SMK)
- ▶ The PKA Key Management Master Key (KMMK)

PKA private keys are protected under two layers of DES encryption. They are encrypted under an Object Protection Key (OPK), which in turn is encrypted under the SMK or KMMK. The OPK is generated for each private key at import time.

### **1.3.2 S/390 PCI Cryptographic Coprocessor (PCICC)**

The S/390 PCICC implements two Master Keys:

- ▶ The DES Master Key, called the Symmetric Master Key
- ▶ The PKA Master Key, called the Asymmetric Master Key

The SMK and KMMK scheme has not been implemented in the PCICC, which has a single PKA Master Key. It is required that the PCICC Asymmetric Master Key has the same value as the CCF SMK, and it is strongly recommended that you set the CCF SMK and KMMK to the same value when at least one PCICC card is also installed in the system. Each PCI Cryptographic Accelerator Feature contains two crypto cards.

### **1.3.3 S/390 PCI Cryptographic Accelerator (PCICA)**

The PCICA is a new cryptographic coprocessor only available on zSeries and requiring z/OS 1.2. This new addition to the mainframe cryptographic hardware is only available on IBM zSeries processors. PCICA is another crypto coprocessor designed specifically for exploitation by SSL. This crypto was designed to extend the scalability of SSL transactions. Each PCI Cryptographic Accelerator Feature contains two crypto cards.

## **1.4 S/390 integrated cryptography implementation**

IBM led the industry by offering the first CMOS cryptographic coprocessor as a standard feature on the S/390 G4, G5/G6 and zSeries servers, and as a priced feature on the G3 server models, in order to meet the increasing needs for data security and integrity.

The S/390 Cryptographic Coprocessor Facility is implemented in CMOS technology on a single chip providing more capability than any previous cryptographic offering. Included in the design are: battery-backed, non-volatile memory storage; laser delete chip personalization; integrated tamper detection and response; high-speed DES, Triple DES, DSS, RSA, Pseudo Random Number Generation; hashing algorithms. Depending on the system model, there may be one or two coprocessor chips in operation.

This hardware runs all current Integrated Cryptographic Feature (ICSF) functions previously offered on ES/9000 @ 9021 processors and includes in addition the Public Key Algorithm (RSA) with digital signature generation and verification.

The software that enables this solution and provides the Application Programming Interface (API) is a follow-up release of the ICSF/MVS product. This release is integrated into the base of OS/390 Version 2 Release 5. This enhanced software release supports new functions built into the chip, and continues to support all previous ICSF/MVS functions found on prior 9021 machines.

OS/390 Version 2 Release 5 also provides support for applications using SET protocol and Visa/MasterCard CVV/CVC card-verification values, including TDES and Double Key MAC. The software supports all the current standards and requirements, as did the previous versions.

Starting June 2000, the PCI Cryptographic Coprocessor (PCICC) is an orderable feature that adds additional cryptographic function and cryptographic performance to G5/G6/zSeries servers. Up to 8 PCICC features may be ordered for a G5/G6, or zSeries server. Each PCICC feature comprises a 4758 Technology-based cryptographic coprocessor card embedded in an adapter package for installing within a G5/G6 zSeries server. Support for PCICC is provided in OS/390 V2R9 by new ICSF functions.

The PCI Cryptographic Coprocessor feature coexists and augments CMOS CCF functions. ICSF transparently routes application requests for cryptographic services to one of the integrated cryptographic engines, either a CCF or a PCICC, depending on performance or requested cryptographic function. For example, RSA signature generation and verification operations (with 1024-bit or shorter keys), such as those typically used by SSL, are routed to both CCF and PCICC engines. On the other hand, RSA Key Generation is performed only on PCICC engines.

PCI Cryptographic Accelerator is the latest cryptographic coprocessor. This new addition to the mainframe cryptographic hardware is only available on IBM zSeries processors. The feature code for the PCICA is 0862. There can be no more than 6 PCICA crypto features per server. PCICA is another adjunct crypto coprocessor designed for exploitation by SSL.

**Note:** The total number of adjunct coprocessors possible on a server cannot exceed eight of any combination of PCICC and PCICA features.

Having one or more of the PCICA features in addition to the CCFs, and one or more PCICCs, will ensure that throughput for SSL-based transactions can be maintained at very high numbers while non-SSL crypto workloads are also

processed. Applications that call ICSF directly for “clear key” RSA operations will also transparently use the zSeries PCI Cryptographic Accelerator Feature.

The PCICA feature supports all public key sizes up to 2048 bits. The PCICA cryptographic hardware feature is designed to perform a very limited set of functions to support SSL cryptographic functions. No data privacy, financial, or key management operations are included in the PCICA design.

### 1.4.1 S/390 integrated cryptography implementation

Figure 1-2 provides a very high-level representation of how the cryptographic coprocessors work in S/390 G5/G6 and zSeries.

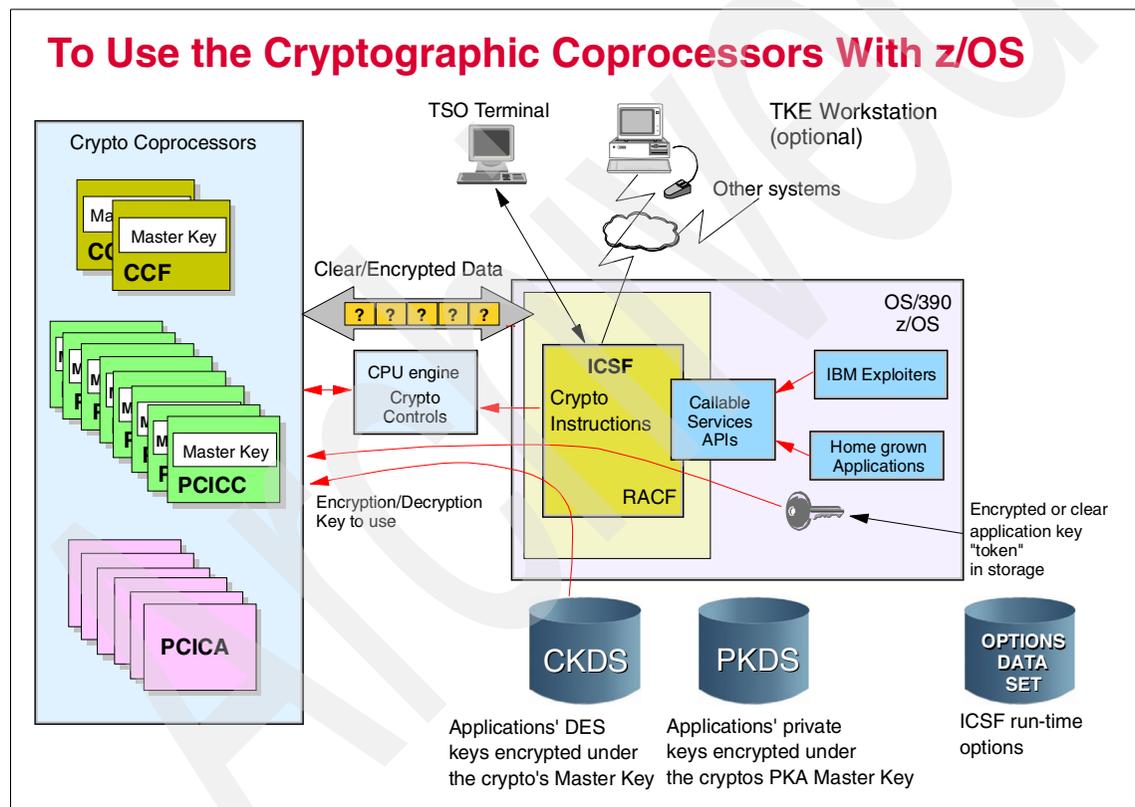


Figure 1-2 S/390 cryptographic coprocessors

- ▶ The exploiters of the cryptographic services call the ICSF API. Some functions will be performed by the ICSF software without invoking the cryptographic coprocessor; other functions will result in ICSF going into routines containing the proprietary S/390 crypto instructions.

- ▶ These instructions are executed by a CPU engine and result in a work request being generated for a cryptographic coprocessor.
- ▶ The crypto coprocessor is provided with the following:
  - Data to encrypt or decrypt from the system memory.
  - The key used to encrypt or decrypt provided by ICSF as per the exploiter's request. Note that these keys are represented as sealed envelopes here, the intent being to stress the fact that these encryption/decryption keys are themselves encrypted and, therefore, unusable when residing outside of the crypto coprocessor.
  - Physically, these keys can be stored in ICSF-managed VSAM data sets and pointed to by the application using the label they are stored under. The Cryptographic Key Data Set (CKDS) is used to store the symmetric keys in their encrypted form, and the Public Key Data Set (PKDS) is used to store the asymmetric keys. The application also has the capability of providing an encrypted encryption key or a clear encryption key directly in memory (that is, to use *as is*) to the coprocessor.
  - For high-speed access to symmetric cryptographic keys, the keys in the CKDS are duplicated into an ICSF-owned data space.

**Note:** Applications using ICSF services can still elect to store keys in non-ICSF key data sets and are, therefore, to manage the keys' security by their own means.

The Trusted Key Entry (TKE) Workstation is an optionally-priced feature. This workstation provides a secure, remote, and flexible method of providing Master Key Part Entry and to remotely manage the cryptographic coprocessors. The algorithm utilizes Digital Signature, Diffie-Hellman, and DES functions to provide a highly secure, auditable, and remote method of key entry, and it can be used by those customers requiring very high security for key entry.

**Note:** A new model of TKE is required to support the PCICC. Differences between this new model and the previous one are described in detail in Chapter 5, "Customizing PCICC and CCF using TKE V3.1" on page 147.

## 1.4.2 Enablement of the cryptographic coprocessors

The S/390 CCF and PCICC are generic cryptographic coprocessors in that they can run various algorithms with different key lengths. However, to conform to export regulations, they are shipped *non-enabled*, meaning that the final user, in the country of use, needs to enable the coprocessors using diskettes provided by

IBM. These diskettes allow activation of the algorithms that the customer requested, and of which use is granted by US and local regulations.

The diskettes need to be loaded only once and are customized so that they load only on the user's machine. The diskettes consist of the following:

- ▶ One enablement diskette for the CCFs installed in the system. The Licensed Internal Code (LIC) in this diskette allows the setting of the CCF Crypto Configuration Controls (CCC) register. This diskette is customized to the CCF chip's serial number.
- ▶ One enablement diskette for the PCICCs installed in the system, for setting the PCICC Function Control Vector (FCV). This diskette is customized to the machine's serial number.

Further details on the cryptographic coprocessors' enablement are given in Chapter 4, "Installation, configuration and startup of ICSF" on page 101.

### 1.4.3 LPAR domains and Trusted Key Entry (TKE)

This section provides an overview of the relationships between logical partitions, physical crypto coprocessors, and domains. It also gives information on defining, controlling and managing these elements.

#### Overview

A cryptographic coprocessor actually has 16 physical sets of Master Key registers, each set belonging to a *domain*. A domain is allocated to a logical partition via the definition of the partition in its image profile; the same domain must also be allocated to the ICSF instance running in the logical partition via the Options Data Set.

Figure 1-3 on page 10 illustrates how logical partitions, physical crypto coprocessors, and domains interact.

## IBM S/390-z900 Cryptographic Coprocessors and PR/SM

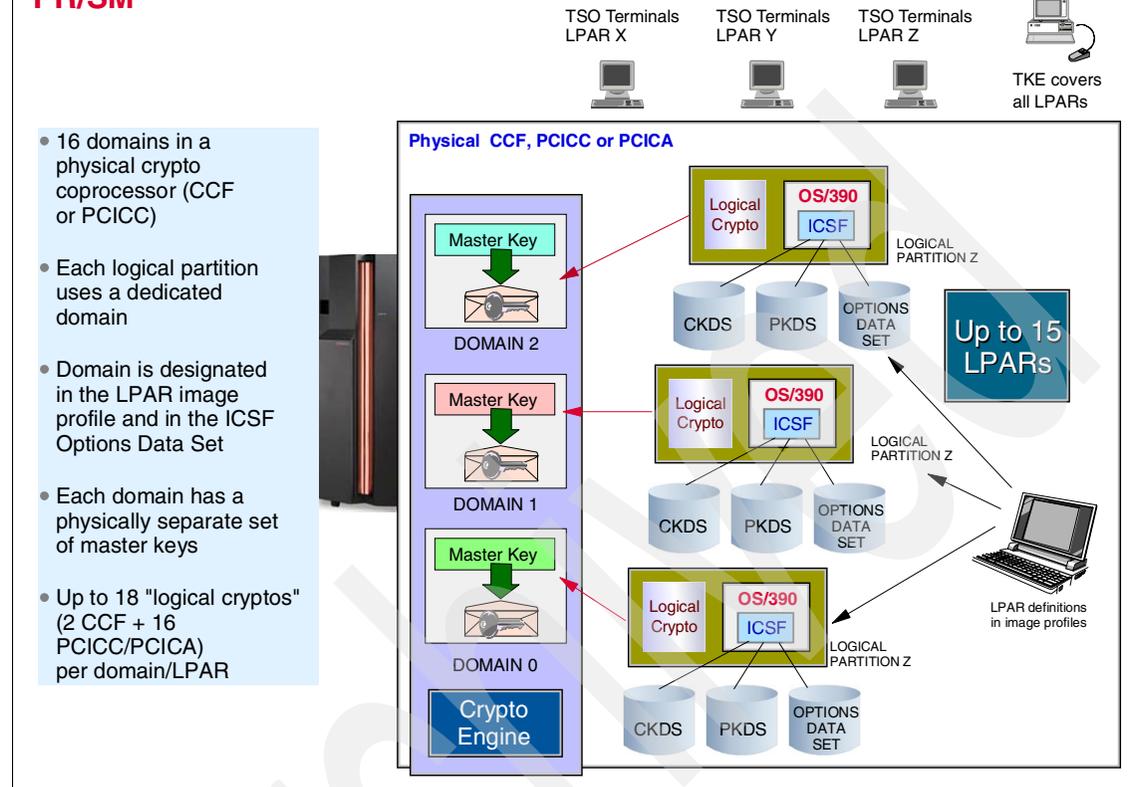


Figure 1-3 Physical coprocessor domains and logical coprocessors

- ▶ Each ICSF instance accesses only the Master Keys corresponding to the domain number specified in its image profile, at the system Service Element, and in its Option Data Set. Each ICSF is seeing a *logical crypto coprocessor* made of the physical cryptographic engines shared among the logical partitions and the unique set of registers (the domain) dedicated to this partition.
- ▶ Each logical partition, or each ICSF instance, therefore has its own unique Master Key values and can encrypt its own set of users' keys with these unique Master Key values, thereby insuring perfect insulation between users of the shared crypto coprocessors from the standpoint of security.

### Defining logical cryptographic coprocessors

The logical cryptographic coprocessors are defined to PR/SM™ by clicking the crypto co-processors shown in the processor view of the image profile. Further

details on the PR/SM definitions required for the cryptographic coprocessors are given in Chapter 4, “Installation, configuration and startup of ICSF” on page 101.

### **Cryptographic controls of the logical partition**

There are controls set up at the logical partition level that influence the scope of the cryptographic functions available from the logical cryptographic coprocessor, as seen from the instance of ICSF running in the partition or from the attached TKE. The set of control parameters available through the logical partition image profile is described in Chapter 4, “Installation, configuration and startup of ICSF” on page 101.

### **Cryptographic services management through the TKE**

The Trusted Key Entry (TKE) Workstation provides a centralized control point for administering the multiple physical or logical crypto coprocessors in a single or multi-system configuration. The TKE communicates with one z/OS instance called the *TKE Host System*.

If the TKE Host System is a logical partition, several domains, corresponding to other partitions in this physical system, can be controlled from the Master Keys’ administration standpoint by a single TKE.

This cross-domain capability is available only when using the TKE Workstation, as opposed to the ICSF ISPF panels. However, ICSF ISPF panels can still be used to administer the local Master Keys, whether a TKE is present in the configuration or not.

## **1.5 Crypto support for z/VM™ and Linux**

The PCICC and PCICA features provide cryptographic support for SSL in Linux environments. The crypto support is only available on G5 and higher servers. For Linux environments, there is no requirement for ICSF. The software communication to the crypto modules within the features is provided via IBM Linux software driver support.

To use a cryptographic coprocessor, the guest operating system has to be able to recognize application requests for cryptography, pass the requests to the cryptographic hardware, and return the results to the application. The z90crypt driver for Linux on zSeries and S/390 exploits the PCICC and PCICA cryptographic hardware for those asymmetric algorithms used by SSL, which results in significant performance improvements for SSL transactions.

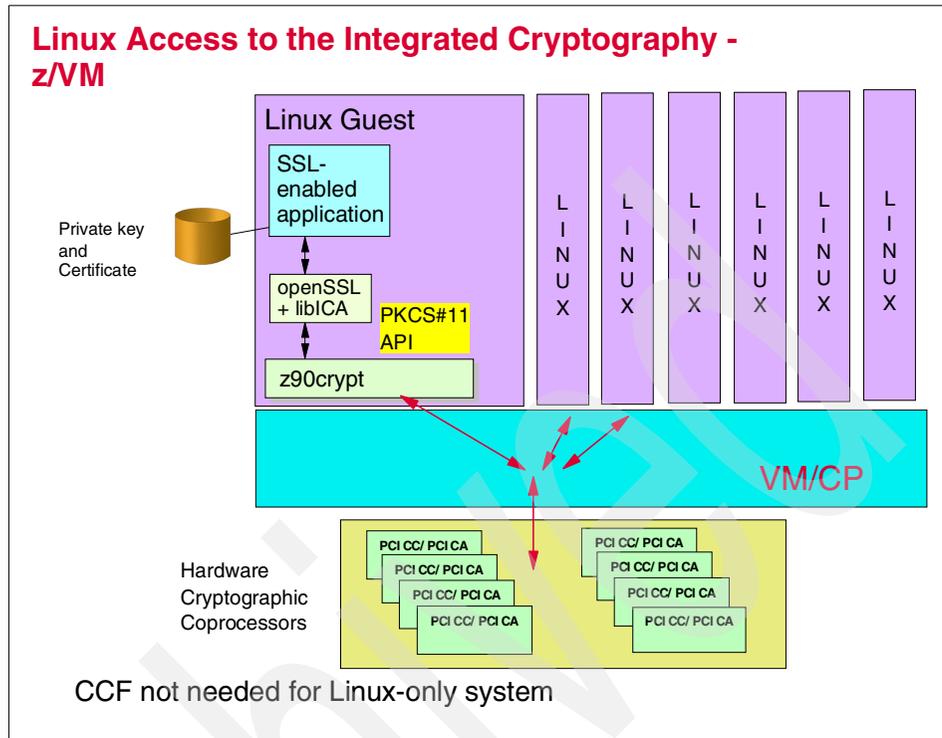


Figure 1-4 Linux access to the Integrated Crypto

The PCICC and PCICA cards may be shared and used by any number of Linux guests. Each operation is discrete and independent of those that precede or follow. z/VM manages the queue of requests so that a guest can see only its own request, as with a shared processor or a shared channel. The Control Program blocks any attempt by a virtual machine to access or change the Master Keys in the PCICC.

## 1.6 Industry standards for cryptographic modules

As the need for computer security grew, the industry needed evaluation criteria for assessing the logical and physical robustness of cryptographic devices. To address this need, the National Institute of Standards and Technology issued the Federal Information Processing Standard (FIPS) 140-1 standard in June 1994. FIPS 140-1 addresses the security requirements of *cryptographic modules*, that is, cryptographic engines, whether in software, firmware or hardware implementation.

This standard covers 11 areas related to the design and implementation of a cryptographic module. Within most areas, a cryptographic module receives a security level rating (1 to 4, from lowest to highest), depending on what requirements are met. For other areas that do not provide for different levels of security, a cryptographic module receives a rating that reflects fulfillment of all of the requirements for that area.

Cryptographic modules that comply with this standard employ cryptographic algorithms, cryptographic key generation algorithms, key distribution techniques, and authentication techniques that have been FIPS-approved for protecting Federal Government unclassified information.

The S/390 CCF hardware single chip cryptographic module has received the highest level of FIPS 140-1 (Level 4) certified by both the National Institute of Standards and Technology (NIST) and the Communications Security Establishment (CSE) of Canada.

A formal list of all FIPS 140-1 products and their associated certification levels can be found at:

<http://csrc.ncsl.nist.gov/cryptval/140-1/1401val.htm>

The S/390 cryptographic coprocessors are physically secure. They provide a tamper-sensing and tamper-responding environment fitting the needs of sensitive applications. Upon detection of physical attack, including penetration, radiation, voltage, excessive cold or heat, the device is “zeroized” and the sensitive information erased.

Archived

## PCICC and PCICA product overview

In this chapter, we provide an overview of the PCI Cryptographic Coprocessor (PCICC) and the PCI Cryptographic Accelerator adapter (PCICA) for z800 and z900 servers. We briefly describe the IBM 4758 Technology as used in the S/390 PCICC implementation starting with 9672 G5/G6 systems, as well as the cryptographic coprocessor enablement concept and the enhanced TKE workstation feature.

## 2.1 Description of hardware

In this section, we explain how the cryptographic coprocessors are implemented in zSeries systems.

### 2.1.1 Definitions

The following definitions will introduce you to S/390 cryptography concepts.

▶ **Cryptographic Coprocessor Facility (CCF)**

This refers to the CMOS cryptographic coprocessor feature logic attached to a central processor on 9672 or zSeries systems. There are one or two CCFs per system, and each CCF has 16 domains. Each domain holds the following set of Master Keys:

- DES Master Key
- PKA KMMK key management MK
- PKA SMK signature MK

▶ **Peripheral Component Interconnect Cryptographic Coprocessor (PCICC)**

zSeries servers also support the optional PCI Cryptographic Coprocessor (PCICC) feature to supplement the standard CMOS Cryptographic Coprocessors, with added functions and performance.

A zSeries PCICC feature includes a pair of PCI Cryptographic Coprocessors built around IBM 4758 card technology, embedded in a Self-Timed Interface (STI)-attached controller card to allow STI-to-PCI interface bus conversion and PCI controller functions. The PCICC installed on zSeries is equivalent of two S/390 G5/G6 PCICC features.

▶ **Peripheral Component Interconnect Cryptographic Accelerator (PCICA)**

The IBM PCI Cryptographic Accelerator is a new unique crypto card designed to implement SSL encryption on zSeries servers.

The IBM PCI Cryptographic Accelerator (PCICA) is a very fast cryptographic processor designed to provide leading edge performance of the complex RSA crypto operations used in the SSL (Secure Sockets Layer) protocol. It has a design point that is different than the existing zSeries CMOS Cryptographic Coprocessor (CCF) and zSeries PCI Cryptographic Coprocessor Feature (PCICC).

### 2.1.2 Hardware implementation

In the following section, we detail the crypto hardware implementation on z900 and z800.

## Crypto hardware implementation on z900

On z900, Cryptographic Coprocessor Facility (CCF) hardware is a standard feature, while PCICC and PCICA are optional features. With the z900, the two cryptographic coprocessors' single chip modules (SCMs) have been moved from the MCM to the CPC cage. The SCMs are plugged directly into the rear of the CPC backplane on all the z900 models.

Note the following:

- ▶ Up to eight PCICC features can be plugged in the z900 I/O cage (FC 2023 cage).
- ▶ Each PCICA feature consists of two cards.
- ▶ Each PCICC card or PCICA card contains two engines and is assigned two CHPID numbers.

### CCF

There are several security levels for the enablement of CCF. The highest level of security is provided with feature code 0875 (Triple DES/DES with PKA and TKE).

Figure 2-1 indicates the physical location of both crypto SCMs (CCFs) on the rear side of the CEC cage.

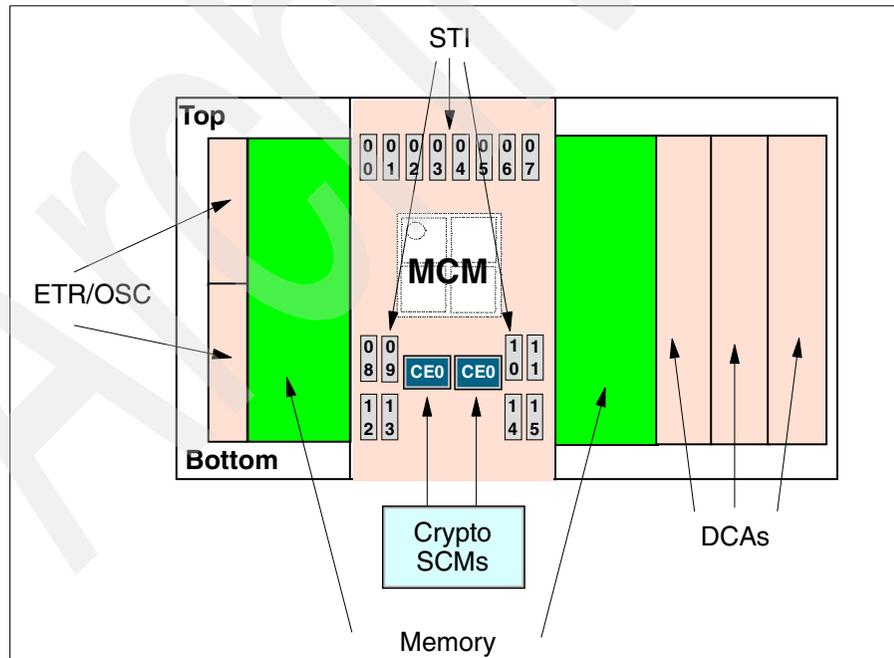


Figure 2-1 z900 rear side showing CCF SCMs

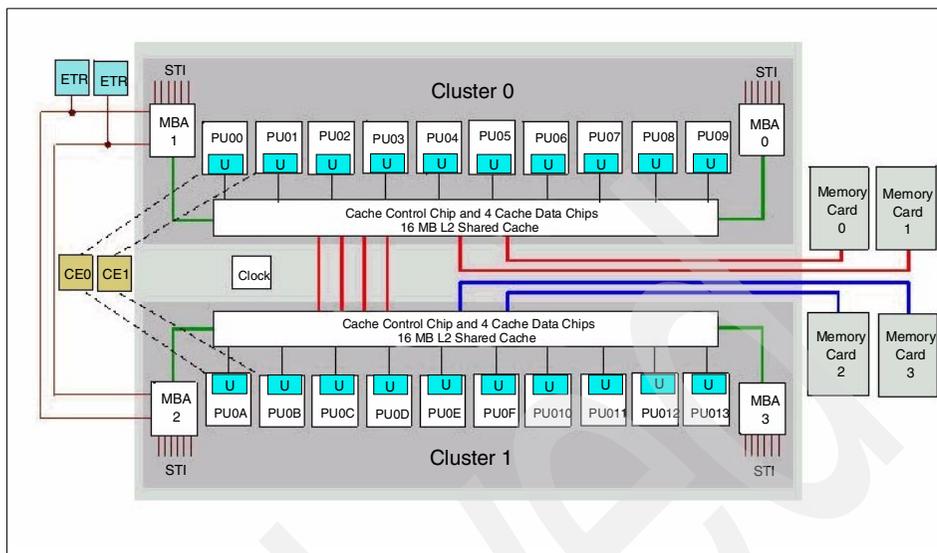


Figure 2-2 20-PU MCM models system structure

Figure 2-2 shows a 20-PU MCM system structure with the two CCF modules referenced as CE0 and CE1 (respectively, Crypto Element 0 and 1). Both Crypto Elements have dual paths to a PU in each cluster on a twin-tailed configuration. This allows continued crypto operation even if a failed PU connected to a CE is spared.

In this case, the recovery of a CE is done by the operating system; that is, the operating system reschedules and dispatches the failed instruction on the same CE via the spare CP. Single CP models only use one CE (while in single CP mode), and after the next power-on Reset (POR), then the spare CE will be used.

**Note:** If a primary CP is not available at IML, the CE will be configured with its associated alternate PU.

### **PCICC/PCICA**

The Crypto coprocessor facility (CCF) is a prerequisite for the PCICC and PCICA only for z/OS.

As of today, there is only one security level for the enablement of PCICC. This security level is provided with feature code 0865 (Triple DES/DES with PKA and TKE).

### ***STI and CHPID plugging rules overview on zSeries***

The z900 I/O cage can house up to seven I/O domains. Each I/O domain may have up to 4 I/O slots, making a total of 28 I/O slots per cage for z900.

Each I/O domain is connected to the CPC via a Self-Timed Interface - Multiplexer (STI-M) card and a 1 GB/s STI cable. Seven STI-M cards (z900) and STI cables are needed to support a full I/O cage.

The following channel types are supported in the I/O cage:

- ▶ ESCON®
- ▶ FICON™
- ▶ OSA-Express
- ▶ PCICC
- ▶ PCICA

Each PCICC or PCICA book occupies an I/O slot in a z900 I/O cage (FC2023).

Figure 2-3 on page 20 shows a PCICC and a PCICA book plugged as a part of the channel packaging on a z900 I/O cage. Refer to 3.2, “Feature codes” on page 53 for a description of feature codes.

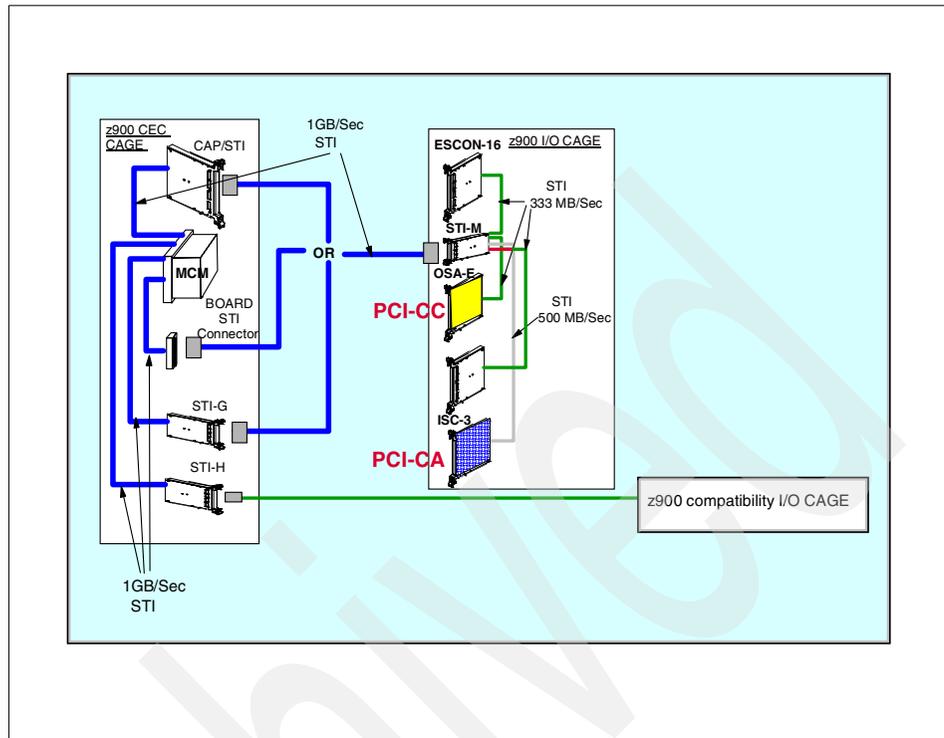


Figure 2-3 Z900 channel packaging overview

## Cryptography hardware implementation on z800

In this section, we detail the crypto hardware implementation on z800.

### CCF

On z800, Cryptographic Coprocessor Facility (CCF) hardware, PCICC, and PCICA are *optional* features. With the z800, up to two optional cryptographic coprocessors single-chip modules (SCMs) can be installed on the Based Processing Unit - Package (BPU-PK). Figure 2-4 on page 21 shows the location of the crypto modules on the BPU-PK card.

As with z900 models, CCFs on z800 have the twin-tailed configuration, allowing continued crypto operation if the failed processor connected to a crypto element is spared.

The security levels for the enablement of the cryptographic coprocessor are the same as for z900; refer to 3.2, “Feature codes” on page 53 for more information.

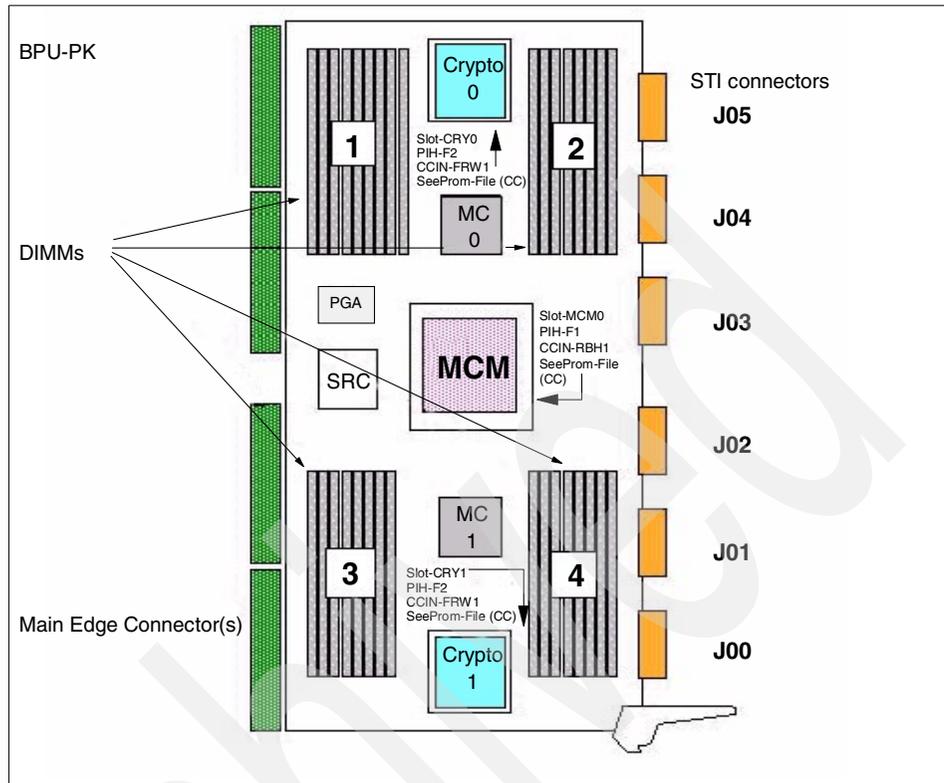


Figure 2-4 z800 BPU processor card with two pluggable Crypto modules

### **PCICC and PCICA**

Note the following plugging rules for z800.

### **STI and CHPID plugging rules overview on zSeries z800**

The z800 I/O cage can house up to four I/O domains. Each I/O domain is made up of up to four I/O slots, making a total of 16 I/O slots for z800. Each I/O domain is connected to the CPC via a Self-Timed Interface - Multiplexer (STI-M) card and a 1 GB/s STI cable. Four STI-M cards (z800) and STI cables are needed to support the full I/O cage.

The following channel types are supported in the I/O cage:

- ▶ ESCON
- ▶ FICON
- ▶ OSA-Express
- ▶ PCICC
- ▶ PCICA

## 2.1.3 Introduction to the S/390 PCI Cryptographic Coprocessors

When ordered on a 9672 G5/G6 or on a zSeries system running OS/390 or z/OS, the PCI crypto coprocessors are always considered a *complement* to the CCF—in fact, the ICSF software requires, as a prerequisite, at least one CCF installed in the CEC in order for the PCI cryptographic coprocessors to be functional.

On zSeries servers, the number of PCICC features cannot exceed eight, while the number of PCICA features is limited to six. PCI crypto coprocessors and PCI crypto engines cannot exceed sixteen.

From a system implementation perspective, a PCICC or PCICA crypto coprocessor unit is an Adjunct Processor (AP). Note that AP designates the processor, while AP ID specifies the number associated with it. The number of APs is limited to 16 on zSeries, and a PCICC or a PCICA is therefore given an AP number between 0 and 15.

### ***PCICC type***

The IBM PCI Cryptographic Coprocessor feature coexists with the CCF, but it also provides additional cryptographic processing power as well as additional capabilities (such as RSA Key Generation) above and beyond the CCF.

The PCI crypto coprocessor unit has 16 cryptographic domains, each holding a set of the following Master Keys:

- ▶ One symmetric Master Key (SYM-MK)
- ▶ One PKA Master Key (ASYM-MK)

The PCICC provides two major advantages over today's Crypto Coprocessor Facility:

1. You can improve cryptographic throughput by simply adding more PCICC cards.
2. IBM allows the programmable implementation of different cryptographic algorithms on PCICC (the CCF currently does not offer that capability).

This answers the need to quickly implement new cryptographic algorithms generated from the banking and industry and is known as User Defined Extensions (UDX) that are detailed in Appendix A, "PCICC User Defined Extensions (UDX)" on page 295.

The S/390 PCICC has been available for G5/G6 CMOS models since June 30, 2000. It uses IBM 4758 technology packaged in an STI-attached controller card (the PCICC card assembly) with a longer battery life, and it delivers signals at a different voltage level (3.3 volt) than the IBM 4758-2 card.

From a functional standpoint, the PCICC assembly also implements the concept of cryptographic “domains”, which is not relevant to the workstation version of the 4758 cryptographic coprocessor.

With the introduction of the PCICC into the S/390 G5/G6 systems, the TKE workstation has been enhanced to allow remote administration of the PCICC cards, while continuing to support the CCF. This V3.1 TKE Workstation includes a 4758-2 cryptographic adapter card for its own cryptographic needs, as a replacement to the former 4755 cryptographic adapter. See r Chapter 5, “Customizing PCICC and CCF using TKE V3.1” on page 147 for details on the utilization of the TKE Workstation.

Another concept introduced to S/390 cryptography by the PCICC is the *retained key*, which is an RSA private key generated by the PCICC upon request of the user. The retained key is kept inside the secure area of the coprocessor and is never to leave the secure area of the coprocessor. It is therefore physically tied to the PCICC it has been generated in and is *never* backed up.

The security level for the PCICC must match the security level of the CCFs. For example, if CCF is using Triple DES with PKA and TKE (Feature code 0875), then any further PCICC upgrade should order Triple DES with PKA and TKE (Feature code 0865).

Support for PCICC started with OS/390 V2R9 with its ICSF functions. ICSF provides cryptographic service requests automatically balanced among all available suitable cryptographic engines.

**Note:** PCICC cards cannot be ordered with the Linux-only z800 model.

### ***PCICA type***

The PCICA unit is designed specifically for maximum-speed SSL acceleration, rather than for specialized financial applications or for secure long-term storage of keys and secrets. As a result, it does not need the tamper design of the zSeries CMOS CCF and zSeries PCICC Feature.

Each zSeries PCI Cryptographic Accelerator Feature contains two cryptographic accelerator cards and can support up to 2100 SSL handshakes/sec. However, the maximum number of SSL handshakes/sec that can be supported on a zSeries server by any combination of CMOS crypto, PCICC crypto, and PCICA crypto is limited by the amount of CPU cycles available to perform the software portion of the SSL handshake.

Unlike the IBM 4758-2 cryptographic coprocessor, there is no microprocessor subsystem (CPU, DRAM, Flash, and so on). The overall operation control, including command decoding, is implemented in the hardware.

Similar to the PCICC feature, two PCICA units are embedded in a Self-Timed Interface (STI)-attached controller card for installing within a zSeries server. The STI-attached controller package allows STI-to-PCI interface bus conversion, and also provides for PCI controller functions.

The accelerator will use the same CHPID type of PCI-Crypto as is used by the PCICC cards. The customer will see it as an Adjunct Processor (AP).

The crypto accelerator also has the following differences from the PCICC feature:

- ▶ It does not support zeroize.
- ▶ It does not support the Random Number Generator function.
- ▶ It does not use a FCV.
- ▶ It does not have tampering detection.
- ▶ It does not have an intrusion latch or battery.
- ▶ It does not support the new UDX functions.

From a system implementation perspective, the PCICA unit is a PCICC unit, an Adjunct Processor (AP).

Starting with z/OS V1R2, ICSF and system SSL functions will automatically use PCICA processors, if available. Usage includes: z/OS HTTP server, WebSphere, TN3270 server, LDAP server, and the CICS® Transaction Gateway server, which are all applications that call ICSF directly for clear key RSA operations.

Linux for zSeries will also support the PCICA card for SSL usage. This applies whether the Linux-only model is used, or the Integrated Facility for Linux (IFL) on a general-purpose model is used, or if Linux is running under a normal CP, or if Linux is under VM.

### **Nature of the PCICC/PCICA features:**

These PCI features, when plugged in the system, are referred to in three ways:

- ▶ PCICC has an eight-character serial number, for reference in a variety of panels and to keep track of the retained keys. This serial number is visible when PCICC status is online/operating (PCICC configured).

The zSeries PCICC feature contains two IBM 4758-based cryptographic coprocessors.

The zSeries PCICA feature contains two IBM cryptographic accelerators.

- ▶ Both types of PCI features have a two-digit Adjunct Processor (AP) number or ID. This number is an index used by the system LIC and ICSF.
- ▶ Because of the way the crypto feature is connected in the system, both crypto coprocessor and accelerator have a CHPID number. The CHPID number is not known nor relevant to ICSF, but is used for hardware management of the feature.

### **PCICC data flow**

In the following section, we explain the PCICC data flow.

#### ***The IBM 4758 Technology and the PCICC hardware data flow***

An IBM crypto coprocessor chip is attached to a 486-class microprocessor, and is mainly used as an accelerator in order to execute the specialized cryptographic functions in dedicated hardware engines (for example, DES and RSA). All the logic shown in Figure 2-5 on page 26 is enclosed in the 4758 physically secure area.

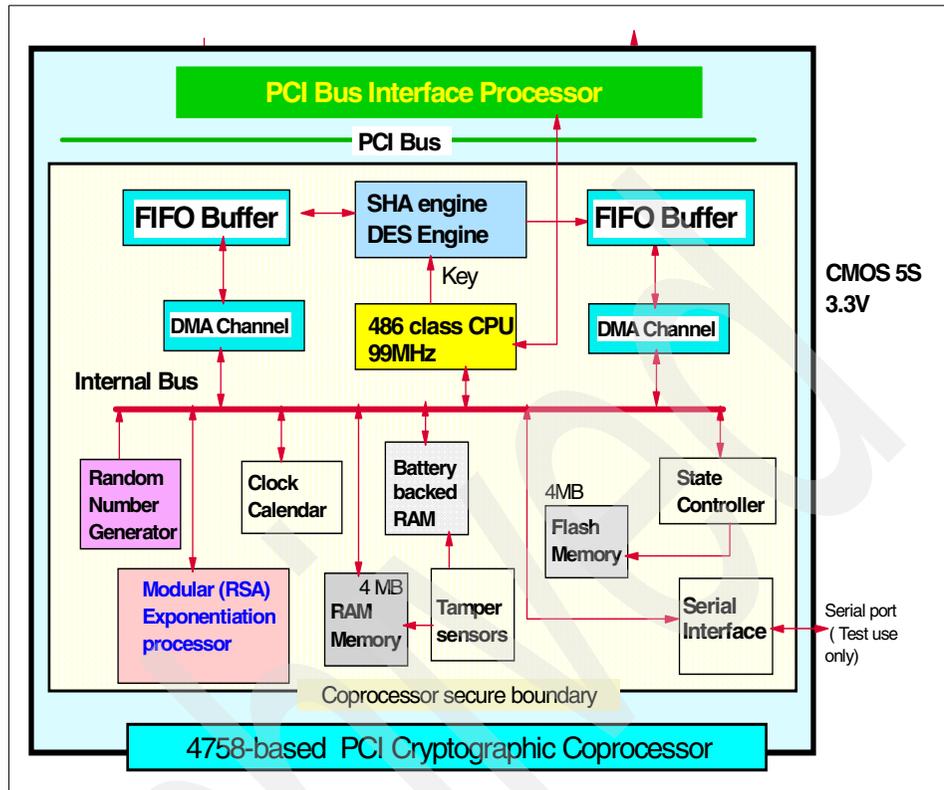


Figure 2-5 4758-based Crypto coprocessor hardware data flow

### The PCICC card

Figure 2-6 on page 27 shows a PCICC card consisting of the assembly of an STI controller. Each feature contains two 4758-based cryptographic coprocessor cards. the PCICC card must *not* be disassembled.

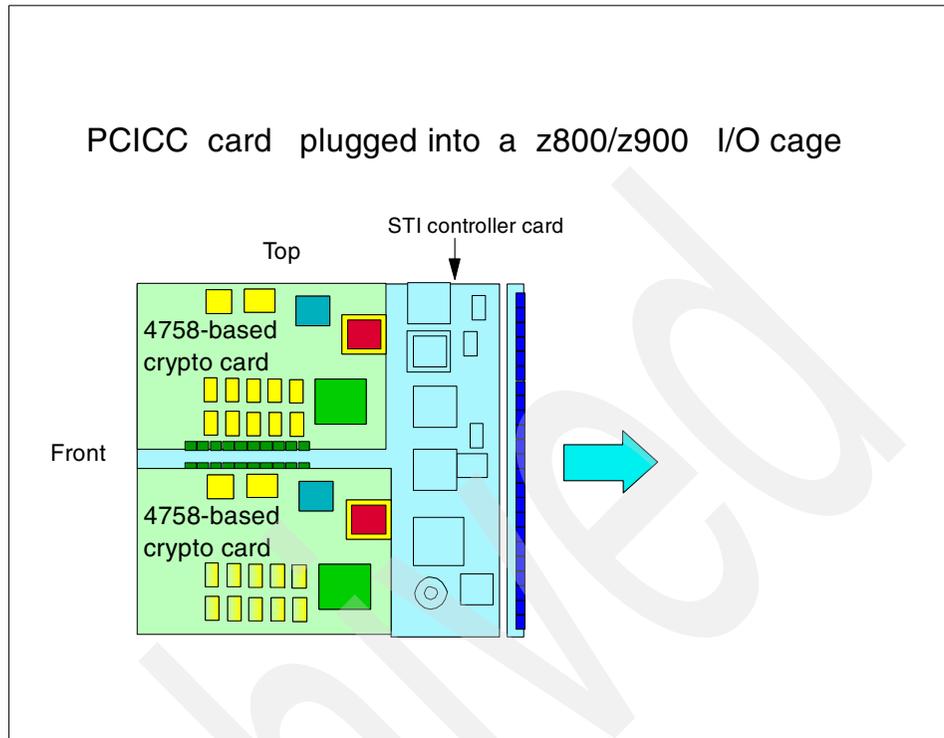


Figure 2-6 PCICC card plugged into z800 or z900 system

Figure 2-7 on page 28 shows a PCICA book consisting of the assembly of an STI controller card and two PCI Crypto Accelerator cards. The PCICA card must *not* be disassembled.

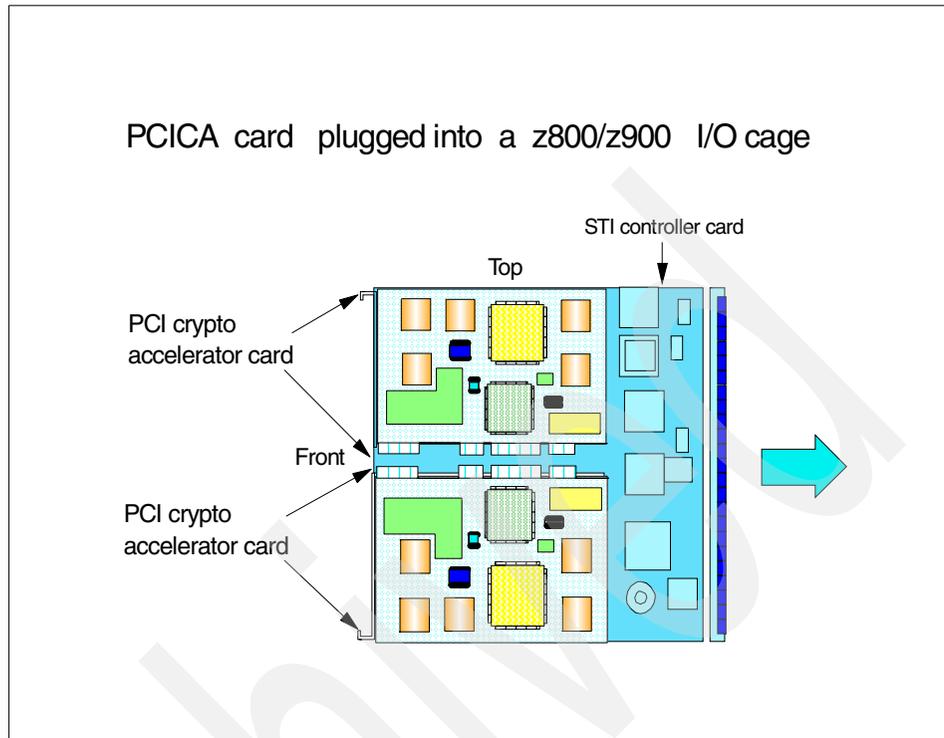


Figure 2-7 PCICA card plugged into z800 or z900 system

### PCICC and PCICA viewing status

Figure 2-8 on page 29 illustrates the PCICC and PCICA end view.

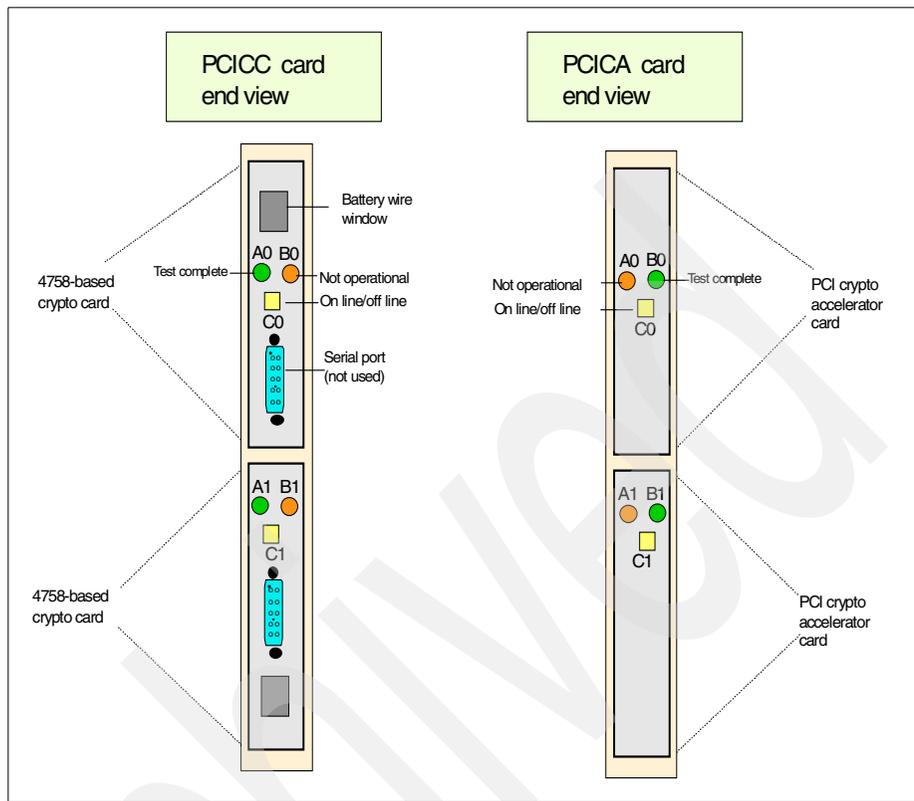


Figure 2-8 zSeries PCICC and PCICA cards: end view

Table 1 lists the meanings of the PCICC card indicators (A) and (B).

Table 1 PCICC and PCICA LEDs (A0/A1 and B0/B1 indicators)

Test complete indicator LED (A) green	Not operational LED (B) amber	Card status
Off	Off	There is no power to the card, or the card processor is in a loop.
Off	Flashing	Diagnostics are running.
Off	On	Status after a card is replaced, from power-on until tests start.
On	Off	The PCI crypto is online (this is the normal status).

Test complete indicator LED (A) green	Not operational LED (B) amber	Card status
Flashing	On	A hardware error is detected.
Flashing	Off	Diagnostics are complete.

Table 2 lists the meanings of the PCICC/PCICA card indicator (C).

Table 2 PCICC and PCICA LED (C) indicator

Online/Offline indicator (C0/C1)	Status
Off	CHPID for crypto adapter is online, and card is communicating with PU.
On	CHPID is offline for maintenance, or an external test is running.
Blinking rapidly	Power-on tests are running.

### PCICA data flow

PCICA data flow is illustrated in Figure 2-9 on page 31. The main components of the PCI Crypto Accelerator card are:

- ▶ Five IBM Ultra Cypher cryptographic engines that perform the following functions:
  - RSA (modular exponentiation) with data key lengths up to 2048 bits
  - Math and special RSA functions up to 2048-bit results
  - DES, T-DES, SHA, and MAC functions
- ▶ Random Number generator
- ▶ Crypto controller module
- ▶ PCI interface module
- ▶ SRAM used as key storage
- ▶ Serial EEPROM used as VPD storage device

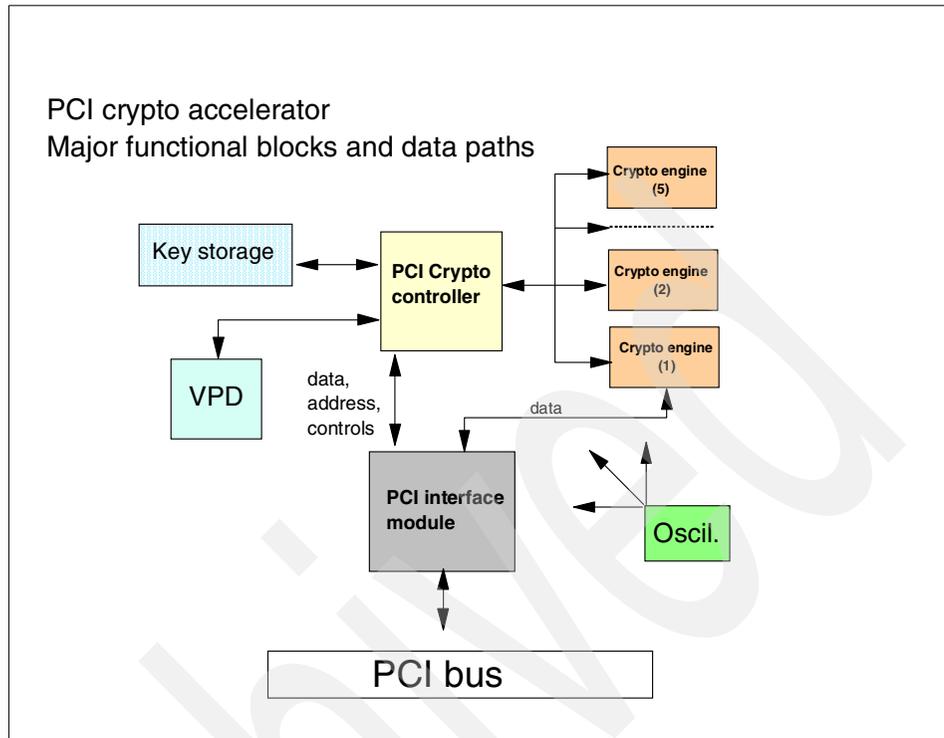


Figure 2-9 PCI crypto accelerator data flow

Major components and their emplacements are shown in Figure 2-10 on page 32.

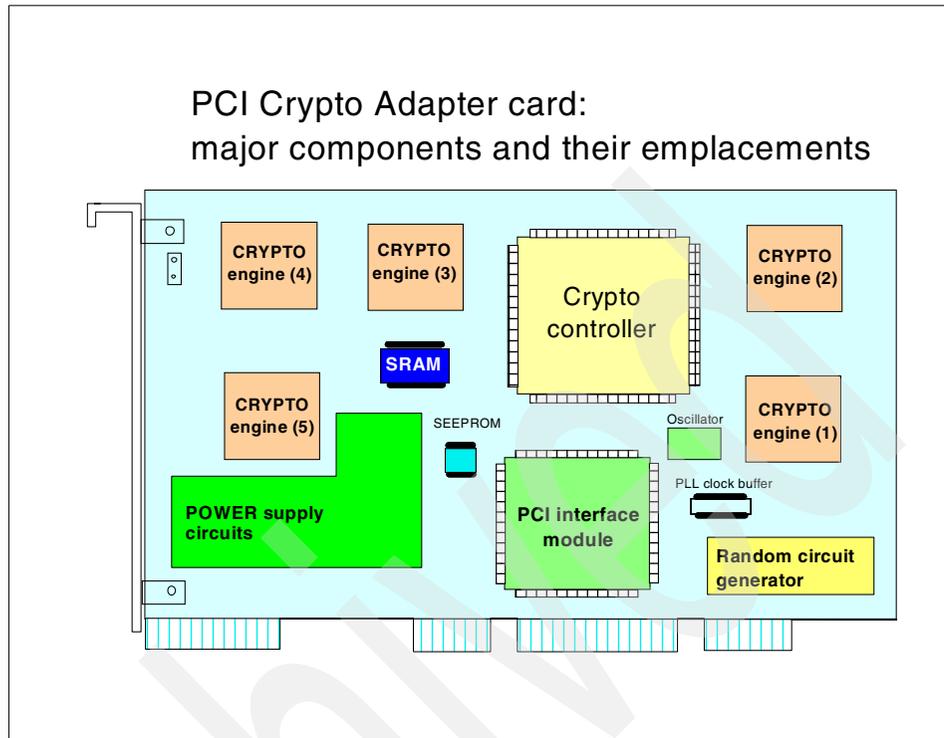


Figure 2-10 Inside the PCI Crypto Adapter card

### 2.1.4 PCICC card: physical security, handling, and shipping

Any attempt to physically penetrate the secure enclosure, or any abnormal modification of the 4758-based card's physical environment, is reported to the card hardware as a "tamper detection". Because the tamper detection sensors are always active in the card as soon as it is manufactured, in the following section we describe the special shipping conditions for the PCICC card.

Figure 2-11 on page 33 contains a longitudinal section which shows the PCICC card internal packaging and the secure enclosure of one of the two 4758-based cards.

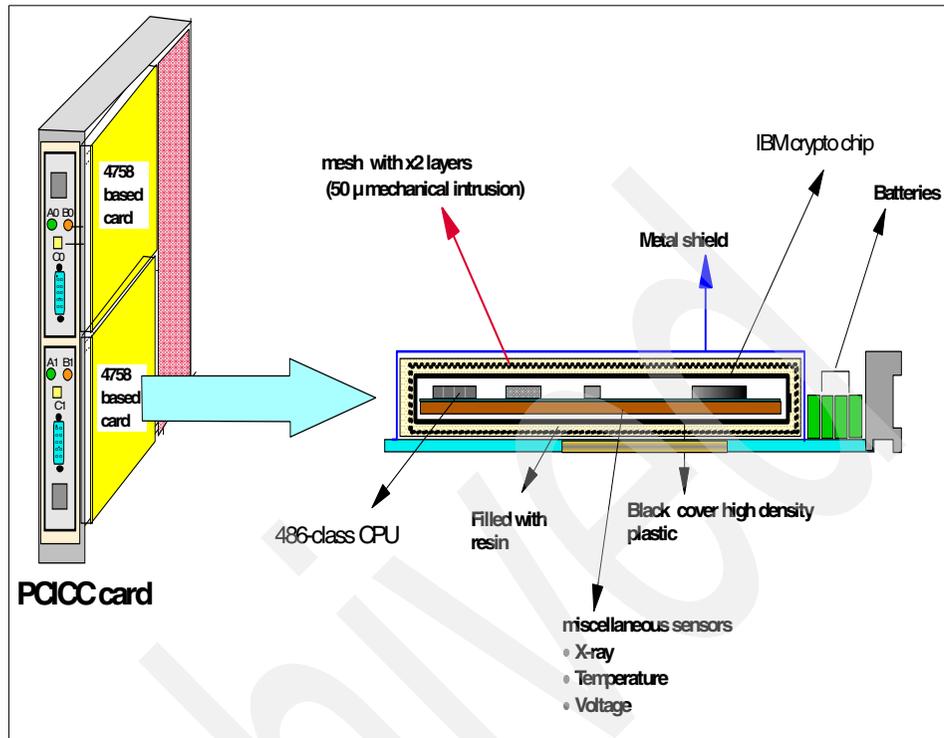


Figure 2-11 4758-based card on S/390: longitudinal section

## Tamper detection conditions

Tamper detection devices are installed to detect physical attempts to obtain the secret values that are stored inside the card. Depending on the abnormal conditions that are sensed, the tamper condition is classified either as *hard* or *soft tamper detection*, with the following results.

### Hard tamper detection

When a hard tamper is detected, the Master Keys and other critical security values for the operation of the PCICC card *are* zeroized by inhibiting the power supply to the card both from the system main power supply and from the on-card batteries. As a result, the card is left in an unrecoverable disable state and has to be replaced. Any of the following events will cause a hard tamper detection:

- ▶ Mesh sensor opens/shorts out
- ▶ X-ray exposure
- ▶ High voltage on +3.3 V ( $3.9 \pm 0.2$  V) or on +12 V ( $14 \text{ V} \pm 0.2$  V)
- ▶ A battery dead condition (VBAT less than 2.54 V)
- ▶ Temperature below  $-20\text{C} \pm 5\text{C}$  or above  $+95\text{C} \pm 5\text{C}$  limits

Special care should be taken to avoid these environmental conditions during card shipment. Refer to “PCICC card shipping conditions” on page 37 for more information.

### ***Soft tamper detection***

When a soft tamper is detected, the card enters an internal reset state. The card appears temporarily unavailable to all external requests when in this state. Once the soft tamper condition is removed, the internal reset condition is dropped and the card becomes usable again. The keys and secret values are *not* zeroized on a soft tamper condition.

Any of the following events will cause an internal card reset:

- ▶ Low voltage detection on +3.3 V ( $2.9\text{ V} \pm 0.1\text{ V}$ )
- ▶ Low voltage detection on +12V ( $10.5\text{ V} \pm 0.15\text{ V}$ )
- ▶ Crypto module internal operating temperature out of window limits ( $0\text{C} \pm 2\text{C} / 75\text{C} \pm 3\text{C}$ )
- ▶ An attempt to load unauthorized code in the PCICC

### **System-detected conditions**

Either of these events will cause a warning at the system console issued by the system’s Licensed Internal Code. In the following sections, we explain these events in detail:

- ▶ An intrusion latch trip
- ▶ A battery low condition (VBAT less than 2.70 V)

### ***Intrusion latch***

The *intrusion latch* is part of the 4758-based card logic and is used to detect when the card is unplugged from the system board. On a subsequent replug, the secret values in the card *will* be zeroized. Note the following points:

- ▶ When the PCICC card is removed from the system board, the intrusion latch is set.
- ▶ The CCA code that runs on the crypto microprocessor checks and resets the intrusion latch at each card power-on.
  - If it finds that the intrusion latch *has* been set since the last card power-on, the code will zeroize the secret values.
  - If the intrusion latch *has not* been set, then the code proceeds with the card initialization without zeroizing the secret values.
- ▶ The intrusion latch will not be set if the cage is moved and the PCICC remains plugged into the cage.

### Notes:

- ▶ The PCICC card should be pulled out of the system I/O cage only if necessary. If the PCICC card is unplugged for any reason (for example, for maintenance or CHPID MES upgrades), the customer *must* be present to reinstall the keys back and the TKE configuration data, if needed, into the PCICC card.
- ▶ The intrusion latch mechanism will also trigger card zeroizing if one of the 4758-based cards is pulled out of the PCICC assembly and is, for instance, re-installed in a workstation.
- ▶ All retained RSA private keys are permanently lost when the PCICC is zeroized; there is, by definition, no backup capability for a retained key.

### **A battery low condition**

Because the PCICC card contains enough batteries to last for the expected card life (8 to 10 years, under normal usage), a battery low condition should normally not occur. The S/390 Licensed Internal Code monitors the low battery bit, so if this unexpected condition is detected, a timely warning is issued to the operator.

A service action should then be taken relatively soon to replace the PCICC card assembly Field Replacement Unit (FRU), as the card will likely fail and become permanently disabled in two to four weeks if the FRU is not replaced.

**Important:** In case the PCICC card must be replaced, the customer *must* be present to reinstall the cryptographic keys in both 4758-based cards.

In addition, if a TKE Workstation has been used to define roles and profiles for the removed PCICC card, then the two new PCICC modules must be recognized and accepted, and the roles/profiles definitions must be reentered from the TKE and sent to the new PCICC modules. Refer to Chapter 5, “Customizing PCICC and CCF using TKE V3.1” on page 147 for details.

### **Zeroizing**

Zeroizing the card results in the following values being rendered permanently useless:

- ▶ The SYM-MK and ASYM-MK Master Keys, in all domains.
- ▶ The users’ retained keys, in all domains.
- ▶ All the roles and profiles defined. The DEFAULT role is reset to the initial state. (Roles and profiles are further explained in 5.3, “TKE application: managing host Crypto coprocessors” on page 170.)

Note that, as for the CCF, you must differentiate coprocessor zeroizing from domain zeroizing:

- ▶ *Coprocessor zeroize* is either invoked from the system Support Element cryptographic controls, or is automatically triggered when the card detects during power-on that the intrusion latch has been set.
- ▶ *Domain zeroize*, in contrast, can be invoked from the TKE Workstation only, and has its effects limited to the selected domain, as explained in “PCICC notebook DOMAINS page” on page 207.

**Important:** There are two buttons in the Support Element panel for zeroizing the PCICC:

- ▶ One button is for zeroizing an individual PCICC card.
- ▶ The other button is for zeroizing *all* PCICC cards.

Be aware that choosing to zeroize all PCICC cards also deletes the Function Control Vector (FCV) in the Hardware System Area (HSA)—therefore, a subsequent attempt to initialize a PCICC card will put the card in check stop state, as there is no FCV to be loaded from HSA.

### Cutting the card disable wire loop

Normally when a PCICC card is to be returned to IBM due to a hardware problem, an upgrade, or for some other reason, the customer’s keys are zeroized via a Support Element panel before the card is removed.

To assure customers that keys have actually been destroyed and that the card has been disabled before IBM removes it from their premises, the card disable wire loop can be cut by service personnel to ensure destruction of the secret values by removing all power to the card.

**Note:** Cutting this wire loop not only destroys the keys, but also *permanently* disables the card.

The wire loop can be accessed by removing a sticker on the tailstock of the PCICC card and reaching through a small hole in the tailstock with a pair of wire cutters. This wire loop is white, about a half-inch long, and is erected perpendicular to the PCICC card printed circuit.

**Note:** IBM support should not perform this procedure unless the customer requests it because Engineering will not be able to perform failure analysis after the wire is cut.

## **PCICC card shipping conditions**

The PCICC card is shipped in such a way that the temperature-sensitive resistors inside the card are not exposed to less than 5 degrees F (-15C). The card should also not be exposed to above 80% humidity, as condensation may cause tamper detection circuits to falsely trip. Each PCICC card is shipped in one large thermal container with five pounds of thermal gel inside the container.

## **2.2 Adjunct Processor (AP) management**

In this section, we develop the concept of the Adjunct Processor as originally implemented in the 9672 G5/G6 and the related system management tasks.

### **2.2.1 Introduction to Adjunct Processor architecture**

The system sees the PCI unit as an “Adjunct Processor”, that is, a coprocessor reachable through the STI cable. Therefore, each PCI unit 4758-based card installed in the system is assigned an AP number between 0 and 15. The system also establishes the correspondence between the AP number assigned to a card and its PC-CC card serial number.

An AP is message-driven and operates asynchronously with respect to the CPUs that invoke it. It is conceptually a queue server handling requests from any client CPU. Figure 2-12 on page 38 is an high-level representation of the AP implementation model.

Cryptographic work to be done is enqueued to a selected AP queue, and then, from a queue, a command request message is sent to an AP. In return, a command reply message is received from an AP.

An AP queue provides accessibility to a given AP from all CPs in the configuration, as opposed to the CCF implementation, where the CCF is physically attached to a central processor. There is a separate AP queue for each domain in domain within an AP, and the AP queues reside in Hardware System Area (HSA) storage.

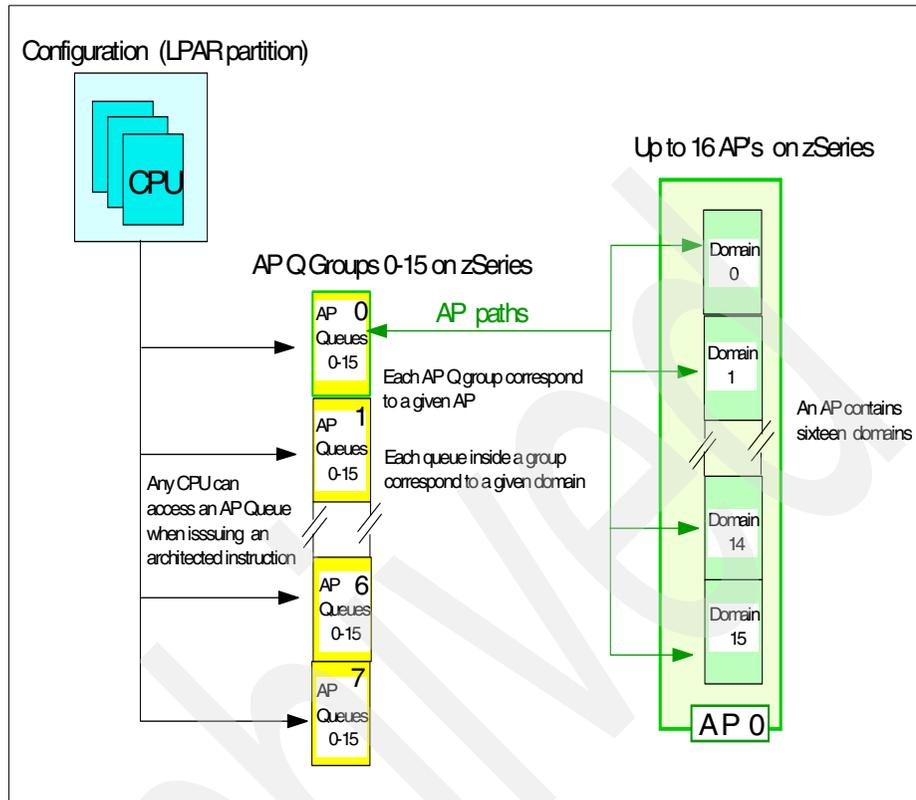


Figure 2-12 AP architecture: high-level view

## 2.2.2 AP management and PCICC initialization

In the following sections, we explain the relationship between an AP and a CHPID number, AP number assignment, and how and why to move PCICC cards.

### AP and CHPID number

For internal system management purposes, an AP is also tied to a CHPID number that is automatically attributed to the PCICC card at installation. Therefore, from a system management standpoint, a PCICC is also a new CHPID type, and this requires the typical changes (LPAR, Advanced Facilities, and so on) scattered throughout the Support Element.

As a consequence, a PCICC card will appear tagged with a CHPID number in the Support Element panels, and can be varied online or offline by using the same facilities as for a normal CHPID

**Note:** z/OS will never see the PCICC as an S/390 CHPID, but only as an AP.

**Important:** Crypto PCIs will *not* be defined in an IOCDS. However, if there are CHPID definition conflicts in the IOCDS (that is, if a CHPID is defined in the IOCDS with a number that matches the CHPID number attributed to the crypto PCI as per the plugging rules), a system Power-on Reset will cause both the already IOCDS-defined CHPID and the crypto PCI to not come online, and an error message will be displayed.

## AP number assignment

AP number assignment for a newly installed PCICC feature card is done at system power-on time or during the PCICC concurrent installation procedure. The AP numbers are assigned to the PCICC cards in sequence, starting with AP 00. From this point on, the AP number is bound to the crypto PCI card serial number (4758-based card).

At system Power-on Reset, the following occurs:

- ▶ The SE (Support Element) passes the cryptographic configuration information to the system Licensed Internal Code (LIC), including the AP IDs-to-CHPID assignment information.
- ▶ The system LIC allocates the Hardware System Area space needed for the maximum possible number of PCICC APs (which is sixteen), whether they are actually installed or not. This will allow the hotplugging of additional cards in the future.

Releasing an AP ID—that is, severing the relationship between an AP number and the PCICC card serial number—can only be done by a manual intervention at the system Support Element. This has to be done when replacing the PCICC card, as an AP/PCICC card affinity should not be changed even if card is moved around in a system. 3.3.4, “Removing one PCICC” on page 90 contains an example of AP ID release.

## Moving PCICC cards

The following sections describe how and why PCICC cards can be moved within a system.

### ***Adding a PCI card (hotplugging or powering off)***

This card addition can take place either while the system is running (hotplugging), or while it is powered off. Note, however, that if the system is powered off, the AP assignment will not occur until the system is powered on. The AP assignment looks for any unassigned AP IDs and assigns one of them to the newly installed card.

### ***Moving a PCICC card within a system***

A PCICC feature card can be moved in the system (and this may possibly be an MES installation requirement) without changing the AP ID-to-PCICC card serial number relationship.

To uninstall a PCICC card, IBM service personnel first use the Nondestructive Hardware Change icon (in the CPC Configuration task) at the system Support Element, then the Remove function. Note that this does *not* free up the AP ID.

### ***Repairing or replacing a PCICC card***

If you repair or replace a PCICC card, the new PCICC card gets assigned the same AP number as the old card.

### ***Moving a PCICC card between systems***

When a PCICC card is removed from the system, either by an MES or as the result of a repair action, service personnel use the AP Manager window on the system Support Element and use the Release function to release the AP ID of the card being removed.

## **2.3 PCICC microcode load**

In the following sections, we describe the organization of the firmware in the 4758-based PCICC, as well as the related installation and system management tasks.

### **2.3.1 The IBM 4758 CCA application**

The IBM 4758 CCA application is a set of firmware which resides in the 4758-based card memory. This firmware implements the CCA functions that can be performed by the card. The CCA application firmware uses the 4758 hardware engines to perform DES, RSA, and other cryptographic functions. The related functions are architected according to a requestor/server model where:

- ▶ The server is in a secure processing environment coprocessor.
- ▶ The major components of the server are a command distributor, command processors, an access control manager, and a Master Key manager.
- ▶ The requests, as served by the card, are atomic units of work.
- ▶ The coprocessor maintains the state of:
  - The Function Control Vector (FCV)
  - The roles and profiles
  - The Master Keys and associated registers
  - The retained keys

The 4758-based card contains firmware to manage its specialized hardware and to control the loading of additional software. The supporting firmware includes the IBM CP/Q++ control program, that is the operating system for the 486 processor, which provides the base for cryptographic application support within the card. The integrity of the card firmware is controlled by digital signature when it is loaded.

The PCICC hardware on G5/G6 supports the secure loading into the PCICC firmware of user-customized extensions to the cryptographic functions provided. Software support to enable this capability will be provided in a future release of zSeries: the User Defined Extensions (UDX) facility can be used to add custom functions to the standard CCA command set. Custom functions are executed inside the secure module of the 4758-based card with the same security as the other CCA functions.

### 2.3.2 The software hierarchy in the coprocessor

As shown in Figure 2-13 on page 42, three segments of microcode need to be loaded into the card:

- ▶ Seg 1, which contains POST1 (the advanced self-test) and Miniboot 1 (the majority of the Miniboot function)
- ▶ Seg 2 CP/Q++, which is empty when shipped from the factory
- ▶ Seg 3 CCA, which is empty when shipped from the factory

Segments must be loaded or reloaded in sequence 1, 2, 3. During Load/Reload, each segment signature is verified. As one segment is reloaded, the higher segment is erased and also has to be reloaded.

**Note:** Seg 0 contains the very basic POST0 and Miniboot 0.

In the PCICC, the card's firmware is reloaded only when there is a mismatch detected between the code and FCV image kept in the system HSA, and the code and FCV currently loaded into the card. The reload can take place only during a system power-on Reset or during a CHPID reset of the PCICC. Note that the card is *not* automatically zeroized when such a reload occurs.

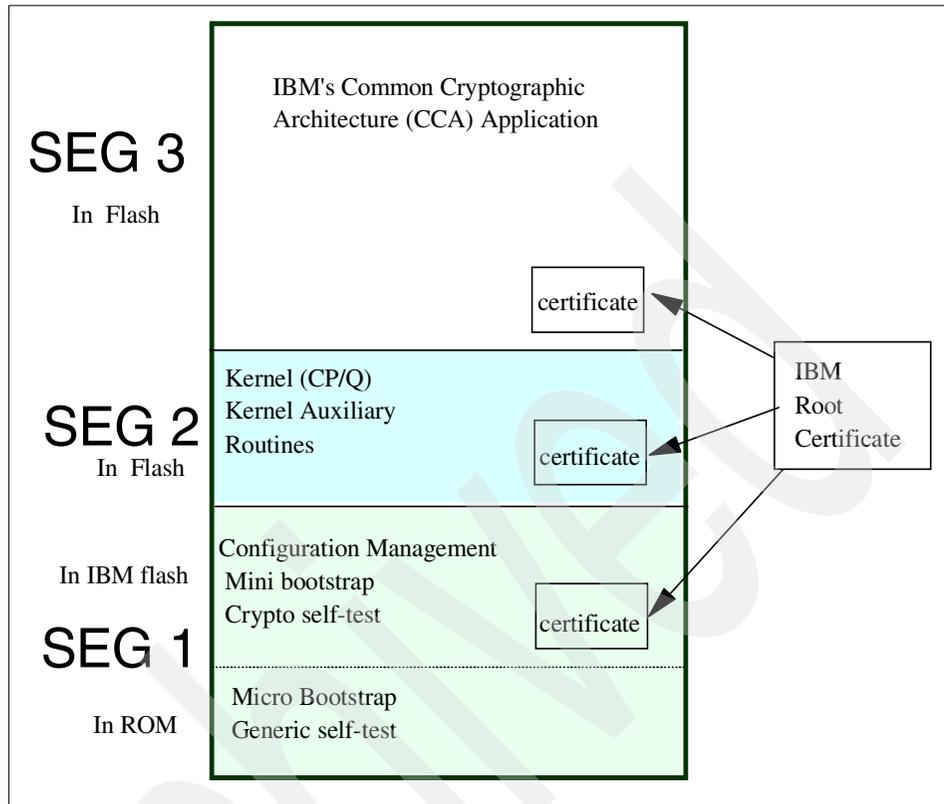


Figure 2-13 Crypto software architecture in the IBM 4758 Technology

### Segment definition

The design for IBM 4758 Technology was motivated by the need to simultaneously satisfy the following requirements:

1. Code must not be loaded into the coprocessor unless IBM has authorized the operation.
2. Once loaded into the coprocessor, the code must not run or accumulate state unless the environment in which it runs is trustworthy.
3. Agents outside the coprocessor that interact with code running on the coprocessor must be able to verify that the code is legitimate and that the coprocessor is authentic and has not been tampered with.
4. The shipment and configuration of coprocessors, their maintenance, and upgrades to the code inside a coprocessor must not require trusted couriers or security officers.

To satisfy these requirements, the card design defines four “segments”: segment 0, segment 1, segment 2, and segment 3, which are described as follows.

▶ Segment 0

This segment is in ROM and contains one portion of “Miniboot”. Miniboot0 is the most privileged software in the coprocessor and, among other things, runs a basic health test of the card and securely loads segment 1, if needed.

▶ Segment 1

This segment is in flash memory and contains the other portion of “Miniboot”. The division of Miniboot into a ROM portion and a Flash portion preserves flexibility (the Flash portion can be changed if necessary), while guaranteeing a basic level of security implemented in the unmodified ROM portion.

Miniboot 1 runs a more extensive health test (POST) on the card and securely loads segment 1 (itself), segment 2, and segment 3, if required.

▶ Segment 2

This segment is in flash memory and contains the 486 operating system with the device drivers that are used to control the cryptographic engines and the PCI bus interface.

▶ Segment 3

This segment is in flash memory and contains the CCA application program.

Detailed information on the 4758 Technology code segments concepts and mechanisms can be found at the IBM 4758 PCI Cryptographic Coprocessor site:

<http://www.ibm.com/security/cryptocards>

You then click the **Library** link.

### **Code-signing key hierarchy**

The code-signing key hierarchy consists of a pair of asymmetric keys assigned to each segment, the principle being that segment level  $n$  verifies the signature of segment level  $n+1$  contents and associated information. This mechanism is used to guarantee the source of the code (that it was developed and signed by IBM-approved people), the integrity of the code executed inside the coprocessor, and that the code is executed on an identified and known coprocessor.

### **2.3.3 PCICC microcode patches**

You can patch PCICC code, when necessary, by using the standard microcode change process (MCL). The microcode changes can affect the 4758-based card microcode itself and/or the STI adapter and controller unit in the PCICC. Such

changes are not expected to be disruptive to the system, because activating a patch requires you to reset the PCICC, which makes it only temporarily unavailable to the applications.

As usual, patch application is concurrent whenever possible. In some special cases, when both the 4758-based card and the STI controller need a microcode update, then the patch is flagged as disruptive and can only be applied with a POR.

Segments 1 to 3 of the PCICC can be Licensed Internal Code (LIC)-updated through the MCL process.

### **2.3.4 Function Control Vector (FCV) enablement**

This section briefly describes the PCICC enablement concept using the FCV diskette. The diskette must be ordered from IBM and installed either by the customer or by IBM service.

As for the CCF enablement diskette, it is intended to enable the cryptographic capabilities of the device while still meeting export and country of use regulations. 3.2, “Feature codes” on page 53 provides information about the feature codes pertaining to the PCICC FCV diskette.

#### **Enablement of the CCF**

Since the original implementation on G3 systems, CCF Crypto Configuration Control (CCC) bits are set according to a Crypto configuration file installed on the system Support Element hard disk. The CCC bits dictate to the CCF hardware which algorithms and key lengths are enabled. The system is shipped out of manufacturing with the CCF CCC bits reset.

The Crypto configuration files are installed by “importing” to the Support Element hard disk the contents of the Crypto Enablement Diskette obtained from IBM. The CCF CCC bits will be set during the system power-on Reset which follows, if the imported file has been selected by the “Select for Next Activation” option on the Support Element.

If the CCF was already enabled and if the just-imported diskette corresponds to a new CCC bits configuration, then the system power-on Reset will set the new value in the CCC bits—but will also zeroize the CCF (that is, Master Keys and authorities will have to be entered again).

Each Crypto configuration file is unique and intended for a specific CCF Crypto Module ID (CMID), which is the number that uniquely identifies a CCF chip. In practice, there is a CCF enablement diskette per system that contains the configuration files applying to the two CCF chips installed on that system.

## Enablement of the PCICC

The equivalent PCICC configuration value is known as the Function Control Vector (FCV). Note that an FCV is not unique to each PCICC, but it *is* unique to the zSeries machine serial number.

The FCV is delivered on a diskette. Because it is specific to the machine serial number, it is common to *all* PCICC cards installed in this system. As a consequence, there is no need to import a new FCV during a PCICC repair or when installing an additional PCICC. As a matter of fact, in contrast to the CCF, a spare PCICC card from the stock will not come with an FCV diskette.

### ***The FCV enablement process***

The FCV enablement process consists of the following steps:

1. The Import FCV function at the Support Element copies the PCICC Crypto configuration files to the Support Element hard disk.
2. The FCV is then uploaded into the system Hardware System Area (HSA), either as a result of a system power-on Reset, or by an option named Load FCV to HSA Immediately at the Support Element.
3. The FCV is downloaded from the HSA into the PCICC cards as a result of either a system power-on Reset or a CHPID reset (that is, by varying a CHPID offline, then online, at the Support Element).

**Important:** Installed PCICC cards go into a check stop state if there is no FCV in HSA when performing a system power-on Reset or a CHPID reset.

See “Import FCV” on page 104 for more details on PCICC enablement operations.

## User Defined Extensions (UDX)

Previous cryptographic products provided the customer with the ability to customize the system's cryptographic code using User Defined Function (UDF) and User Developed Program (UDP). The PCI Crypto on the 9672 G5/G6 and the z900 products have not provided these functions.

For customers who required UDX support, there was an RPQ process but that required a long turnaround time. It also had a perceived concern of security, since the customer's requirements had to be sent to IBM for the implementation.

For GA2 of the z900, UDX support similar to that which was offered on the earlier processors is available.

To utilize UDX, customers must follow these steps:

1. Develop a UDX using the IBM 4758 CCA UDX Application Program Development Toolkit.
2. Create a UDX package file using the zSeries Signer and Packager utilities that run on a separate workstation (that is, Windows NT). The file will contain support to allow the customer to import the UDX into a PCI Cryptographic Coprocessor installed in a z900 server.
3. Copy that UDX file to a diskette.
4. Use the HMC in Single Object Operations or the Support Element to invoke a new Import UDX File task to import the UDX file.
5. Invoke a new Activate UDX support element task targeted for specific APs.

Two new tasks are also provided:

- ▶ Reset UDX to IBM Defaults
- ▶ Query UDX Level function (similar to today's PCI Crypto Read VPD) to display the version/time stamp.

### 2.3.5 Software support of PCICC coprocessors

This section contains a brief description of the changes implemented in OS/390, z/OS and its Integrated Cryptographic Service Facility (ICSF) component to support the PCICC, the PCICA, and the current TKE V3.1 workstation.

#### OS/390 2.9 support for PCICC

Support of the PCI Crypto coprocessor started with OS/390 2.9.

- ▶ The IO Supervisor functions were enhanced to detect and report the change in card condition when the card is being varied online from the Support Element menu.
- ▶ ICSF 2.3 is part of the OS/390 2.9 base. ICSF 2.3 provided the functions required to support the PCI Crypto Coprocessor (PCICC) along with the Crypto Coprocessor Facility (CCF) chip. It also provides the software piece required to establish connectivity between the TKE and the Crypto coprocessors via TCP/IP protocol.

**Note:** Starting with OS/390 2.4, ICSF is an OS/390 “exclusive” product, meaning that it cannot be delivered outside of the base OS/390; therefore ICSF 2.3 should be referred to externally as *OS/390 2.9 ICSF* in order to avoid any confusion.

## ICSF overview

ICSF provides support for the following:

- ▶ The Commercial Data Masking Facility (CDMF), an exportable version of DES cryptography
- ▶ DES and Triple DES encryption, for privacy
- ▶ The transport of data keys through the use of the Rivest-Shamir-Adelman (RSA) public key algorithm
- ▶ The generation and verification of digital signatures through the use of both the RSA and the Digital Signature Standard (DSS) algorithm
- ▶ The generation of RSA and DSS keys
- ▶ The SET Secure Electronic Transaction standard, created by Visa International and MasterCard
- ▶ The PKA Encrypt and PKA Decrypt callable services, that can be used to enhance the security and performance of Secure Sockets Layer (SSL) security protocol applications

### 2.3.6 The TKE V3.1 Workstation

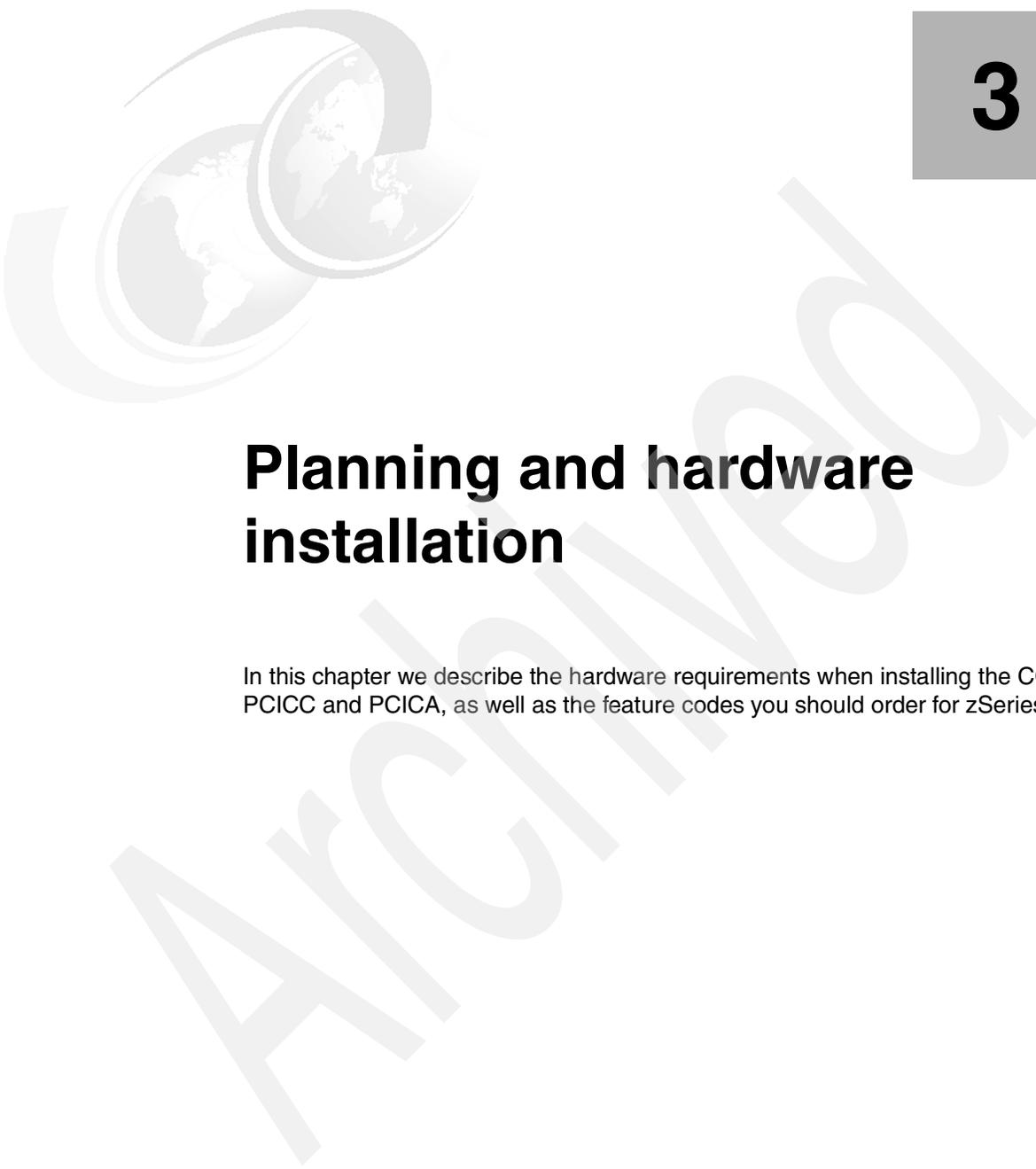
TKE version V3.1 is the workstation version currently required to support the PCICC. Details about this workstation are provided in Chapter 5, “Customizing PCICC and CCF using TKE V3.1” on page 147.

You can check your current TKE level; it is briefly displayed at the bottom of the screen when the TKE application starts.

#### Notes:

- ▶ TKE V3.1 communicates with the z/OS host using TCP/IP.
- ▶ It uses a 4758-2 cryptographic adapter card.
- ▶ The Key Transfer function, which enabled the secure extraction of a Master Key from one system and its transfer to another system, is no longer available.
- ▶ TKE V3.1 does not support a smart card reader.

Archived



## Planning and hardware installation

In this chapter we describe the hardware requirements when installing the CCF, PCICC and PCICA, as well as the feature codes you should order for zSeries.

## 3.1 Hardware requirements

IBM Customer Engineers are responsible for installing the hardware by following the Installation Instructions that are generated for a specific system serial number. These instructions describe the card movements which may be required on the specific system to perform the PCICC/ PCICA card installation, and they are therefore dependent on the accuracy of the system configuration as recorded in the IBM Vital Product Data (VPD) database.

### 3.1.1 Hardware required by product

The following hardware is required.

#### **CCF**

##### ***zSeries 800 server***

The Crypto coprocessor hardware (CCF) is optional on the z800. Two Crypto modules, packaged as SCMs, can be plugged onto the processor card BPU-PK.

The Crypto SCMs are FRUs which can be plugged in the field as part of the MES process. The MES package includes:

- ▶ Two crypto SEEPROM diskettes (one for each crypto SCM identified by the SCM serial number) that will be used to build Vital Product Data (VPD)
- ▶ Two appropriate “Crypto Enablement” diskettes (one diskette for each crypto SCM); refer to Table 3-2 on page 54
- ▶ Two crypto SCMs (Single Chip Module), battery units and cables
- ▶ A tool kit, including insertion tool
- ▶ Installation Instructions

Crypto battery units (CBUs) are mounted on the Power control unit in the CEC cage. They provide power to retain a customer’s Cryptographic keys during a system power down.

Crypto SCMs are also FRUs which can be exchanged in the field as part of a repair action. The Field spare kit will contain one SEEPROM diskette for the new SCM identified by its S/N, as many enablement diskettes as there are supported feature codes (with each diskette containing data for the SCM to be replaced), and a tool kit for SCM removal/installation.

##### ***zSeries 900 server***

The Crypto coprocessor hardware (CCF) is present on the z900. Two Crypto modules, packaged as SCMs, are plugged onto the rear side of the CEC cage.

The CCF is enabled using two appropriate “Crypto Enablement” diskettes, one diskette for each crypto SCM; refer to Table 3-2 on page 54.

Crypto battery units (CBUs) are mounted on the front of Distributed Control Assemblies (DCAs) in the CEC cage. They provide power to retain a customer’s Cryptographic keys during a system power down.

Crypto SCMs can be exchanged in the field as part of a repair action using the same process as described for z800 server. In case of CEC cage replacement, the new spare part from stock will include the cage with both crypto SCMs pre-plugged, the board SEEPROM, the SEEPROMs for the two SCMs and the appropriate crypto enablement diskettes as required.

Table 3-1 lists SEEPROM information for the two SCMs (either z800 or z900).

*Table 3-1 SEEPROM information for the two SCMs*

Diskette Vol ID	Diskette File ID	SEEPROM file on SE hard disk
BBRU_CR\$DAT	BBRE_SHP.SPM diskette for SCM0	D:\SEEPROM\BBRUCRY0.SPM
BBRU_CR\$DAT	BBRE_SHP.SPM diskette for SCM1	D:\SEEPROM\BBRUCRY1.SPM

After loading, the files will be automatically renamed on directory D:\SEEPROM\ as files BBRUCRY0.SPM and BBRUCRY1.SPM.

## **PCICC and PCICA**

Chapter 2, “PCICC and PCICA product overview” on page 15 contains a product overview. The plugging rules for PCI Crypto cards are addressed in I/O Support and STI and Channel Plugging Rules.

The hardware requirements for PCI crypto cards installation on z800 / z900 are as follows:

- ▶ The PCICC feature is only supported on 9672 G5/G6 and zSeries systems.
- ▶ The PCICA feature is only supported on zSeries systems with a minimum level code for z900 only (Support Element Driver code 3C for z900).
- ▶ The maximum number of PCICC/PCICA features that can be installed in a z800 or z900 system is eight.

- ▶ The PCICC feature consists of:
  - The PCICC feature (FC 0861) consists of one card to plug into the system. Note that for zSeries, the PCICC feature contains two IBM 4758-based cryptographic coprocessors
  - An CV diskette (FC 0865) to enable PCICC (only one FCV diskette per machine). The FCV is created for a specific system serial number.
- ▶ The PCICA feature (FC 0862) consists of a hardware PCICA card only (no FCV is required). The zSeries PCICA card contains two IBM cryptographic accelerators.
- ▶ From an ICSF software point of view, the PCICC or the PCICA cannot operate if the CCF itself is not enabled and in operation. Note that PCICA can work without CCF on Linux-only models.
- ▶ The HSA space consumed by the PCICC is 14.5 MB. This value is independent of the number of installed cards.
- ▶ On the z800 Linux only model called 0LF, only the PCICA feature is available.

PCICC and PCICA cards can be exchanged in the field as part of a repair action. The new spare card comes from stock with no diskette.

### **Repair action on CCF/PCICC:**

Repair action on CCF modules is always disruptive (POR needed).

**Important:** When a Service Action requires the removal of the Battery Backup from the cryptographic unit, the Master Keys and configuration data is lost.

This will happen on a z900 system when replacing any of the following:

- ▶ One SCM on the rear side of the CEC cage
- ▶ A PCICC card in a Z900 I/O cage (intrusion latch is the cause)
- ▶ The CEC cage containing the SCMs
- ▶ Both Distributed Control Assemblies (DCAs) in the CEC cage containing the batteries

This will happen on a z800 system when replacing any of the following:

- ▶ SCM on the BPU-PK card
- ▶ A PCICC card in the I/O cage (intrusion latch is the cause)
- ▶ MCM on the BPU-PK card
- ▶ Memory DIMMs on the BPU-PK card
- ▶ The BPU-PK card itself
- ▶ Both DC/DCs power supplies in the CEC cage containing the batteries

As a consequence:

- ▶ Master Keys stored in the SCMs or in the PCICC card will be lost. It is the customer's responsibility, upon completion of the maintenance task, to restore the Master Keys and any existing retain keys in all domains.
- ▶ If a customer is using a TKE Workstation, then the corresponding configuration data (CCF or PCICC) is lost and needs to be restored. It is the customer's responsibility, upon completion of the maintenance task, to restore this data from the TKE for all domains.

For more information, refer to Chapter 4, "Installation, configuration and startup of ICSF" on page 101 and Chapter 5, "Customizing PCICC and CCF using TKE V3.1" on page 147.

## 3.2 Feature codes

See Table 3-2 on page 54 for a summary of CCF, PCICC and PCI-A feature codes.

### **PCICC and PCICA plugging rules for zSeries 2064-z900 and 2066-z800:**

- ▶ Number of PCICC cards plugged: 0 to 8 cards.
- ▶ Number of PCICA cards plugged: 0 to 6 cards.
- ▶ The total of PCICx features cannot exceed 8.

Table 3-2 zSeries crypto features with G5/G6 equivalent (including the latest crypto feature changes of Nov. 2002)

Description	2066-z800 CCF	2064-z900 CCF	z800 / z900 PCI crypto features	9672 G5/G6 equivalent
Crypto Hardware Present (SCMs)	0800	0800		0800
Smart Card Reader	Not available	Not available		0807 or 0867
TKE Ethernet Workstation	0879	0869		0809 or 0869
TKE Token Ring Workstation	0876	0866		0806 or 866
Triple DES with PKA & TKE [+WT sub.]	0875 (two diskettes)	0875 (two diskettes)	0865 (one diskette)	0835 (one diskette)
PCI Crypto cards (hardware only)			0861 (PCICC) dual ports 0862 (PCICA) dual ports	0860 (single port)

Note the following points:

- ▶ The CCF hardware feature (0800) is standard on 9672 G4/G5/G6 and z900 systems, but is an optional feature on z800.
- ▶ The Trusted Key Entry unit (TKE) will continue to be offered as an optional feature, on zSeries and is available in two communication formats: Token Ring and Ethernet. The TKE provides a means by which the customer can further restrict access via coded key entries, thus limiting who may use or operate the system.

Activating feature code 0875 (T-DES with PKA and TKE) means that the CCF Crypto is enable to work with an attached TKE Workstation. If needed, one or two TKE Workstations can be ordered separately (TKE hardware ordering is discussed in the following section).

With T-DES, PKA, and TKE crypto configuration already activated, you will be able to install TKE Workstation concurrently (that is, there is no need to import/select and activate a new crypto configuration).

The TKE hardware itself (TKE V3.1) is orderable by selecting:

- For z900
  - Feature code 0866 (TKE Workstation with a token ring adapter), or feature code 0869 (TKE Workstation with an ethernet adapter)
- For z800
  - Feature code 0876 (TKE Workstation with a token ring adapter), or feature code 0879 (TKE Workstation with an ethernet adapter)
- ▶ Each PCICC card to be installed in the system has feature code 0861.
- ▶ Because CCF enablement is a prerequisite to the installation of a PCICC, feature code 0875 must have been ordered and installed. This is the feature corresponding to the CCF enablement diskette.
- ▶ The PCICC enablement diskette (FCV) is orderable by using feature code 0865 (T-DES with PKA). This feature code must also match the encryption strength ordered for the CCF. As already mentioned, there is no PCICC feature code pertaining to the TKE enablement. Also, you only order the feature code once, since a single FCV diskette covers all possible PCICC cards in a given system serial number.
- ▶ Each PCICA card to be installed in the system has feature code 0862. There is no enablement diskette for PCICA.
  - The PCICC or PCICA card should be plugged using the normal plugging rules. The order increment is 2 engines/1 card.
  - The Smart Card Reader is not offered on z800/z900 systems.

### 3.3 Concurrent PCICC/PCICA installation tasks

In this section we discuss the steps involved in performing PCICC and PCICA installation and activation.

#### Assumptions

- ▶ The system Support Element has already been upgraded to Driver 3G.
- ▶ The CCF cryptographic configuration in effect matches the PCICC intended configuration (as an example, the CCF uses feature code 0875 and the PCICC to install has feature code 0865).

- ▶ The logical partition image profiles have already been set up with the new PCICC page, where the PCICC APs intended to be installed have been entered into the candidate list.
- ▶ The PCICC/PCICA card locations and CHPID numbers are known from the Sales Ordering System the day the order is entered. The report from the configurator contains the relevant data in the CFREPORT/CHPID REPORT, and is normally supplied to the customer for planning purposes.

Using this data, it is possible to check for possible HCD configuration changes and to determine whether the CHPID number required for PCICC /PCICA installation is free.

### ***System references***

We performed the cryptographic products installation on a z900 system (machine type 2064, model 216) located in the Montpellier Products and Solutions Support Center (PSSC). CCF hardware was present on the system but not enabled, and the PCICC/PCICA cards were not installed.

The system was shipped from Manufacturing with a Support Element code at Driver 3G plus the latest MCLs. (This is expected to be the most common case for z800 and z900.)

The system was shipped with the following crypto features codes (refer to Figure 3-1 on page 57):

- ▶ CCF FC 0875 (TDES with PKA and TKE) contained in two enablement diskettes
- ▶ One FC 0861 (PCICC card)
- ▶ One FC 0862 (PCICA card)
- ▶ PCICC FC 0865 (TDES with PKA and TKE) contained in one FCV diskette

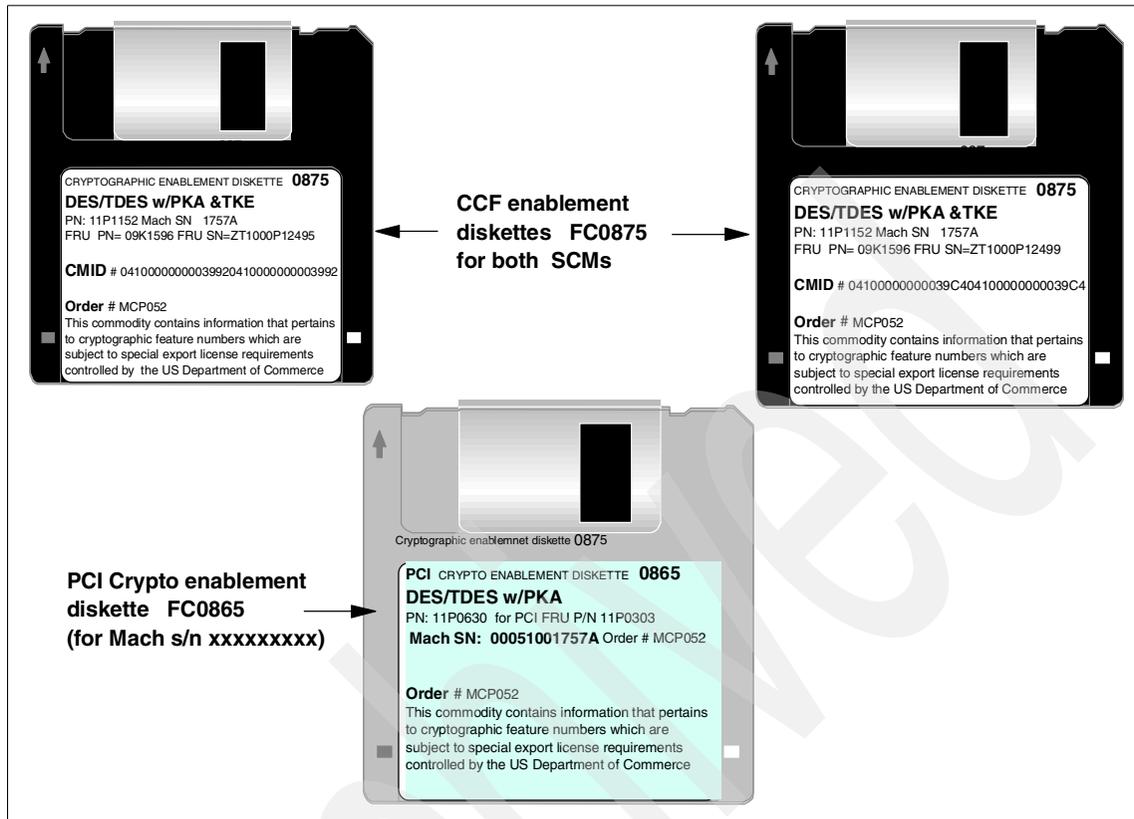


Figure 3-1 Cryptographic enablement diskettes for CCF and PCICC

We used three upgrade scenarios:

1. The first scenario is related to an environment where the customer is using CCF and plans to install and activate PCICC and PCICA concurrently.
2. The second scenario is related to an environment where the customer is using CCF and PCICA and wants to activate PCICC concurrently.
3. The third scenario is related to User Defined Extensions (UDX) file installation on a PCI-crypto coprocessor.

In the following sections, we examine these scenarios in more detail.

### 3.3.1 First scenario

We followed these steps in this upgrade scenario:

1. Enabled CCF, POR, IPL, and started ICSF (IBM support/SysProg).

2. Changed image profiles to define PCI cryptos in the Candidate list (SysProg). There was no PCI crypto defined in the Online list.
3. Activated the image(s) during the next scheduled maintenance slot. Next we IPLed, and then started ICSF using CCF only.
4. Installed PCICC and PCICA cards (IBM support)
5. Configured PCICC and PCICA CHPIDs On (IBM support)
6. Activated PCICC from ICSF (SysProg), then generated Master Keys for PCICC.

### **CCF enablement**

1. We first imported the CCF cryptographic configuration using the two crypto enablement diskettes related to feature code 0875. Then we selected this configuration for the next system activation. (Note that any CCF configuration change requires a POR.)
2. From image profiles, we selected **crypto coprocessors** and **update crypto page**.
3. We performed system POR IOCDS A1 LPAR mode with images S501 and S502.  
  
Since we used image S501 for our residency, we activated image S501.
4. We started ICSF and created DES and PKA Master Keys using Pass Phrase Initialization.

CCF was then ready for crypto applications.

### **Concurrent PCICA/PCICC installation**

With no PCI crypto card plugged on the system, then from the configuration task:

- ▶ The PCI cryptographic management icon is empty.
- ▶ The PCI cryptographic configuration icon is not accessible.

**Note:** This could be seen as disruptive because—before installing the PCICC cards by using the “Nondisruptive Hardware Change” task to import/select and save the FCV—you need to set up the image profiles with the PCI Crypto Candidate list populated, and then activate the partitions. The disruption occurs when the images are activated.

If the image profiles are already set up and activated, with the PCI Crypto Candidate list populated during a previously scheduled slot, then PCI Crypto CHPIDs can be brought online and fully functioning.

### ***Prior to PCICC/PCICA installation***

1. ICSF was running, with Crypto CCF configured.
2. In Sysprog mode, we changed the profiles for both images to include PCICC, so that both images then had AP0 to AP15 (the maximum number of APs allowed) in the candidate list PCI Crypto page (and nothing in the online list).
3. We deactivated both images and activated them again to take the image profile changes into account.
4. We IPLed the S501 image, which was used for the project, and started ICSF.

### ***PCICC/PCICA installation steps - overview***

Following is an overview of this installation process. (In the section that follows, we illustrate these steps and walk you through the details, using Figure 3-2 on page 60 through Figure 3-28 on page 83.)

1. Install the PCICC card in service mode.
2. Import, and then select FCV (TDES) using Load FCV into HSA storage immediately and Save.

At completion, you will see:

- The PCI Coprocessor Management panel, showing chpids 10 and 11 installed and associated to AP0 and 1.
- The PCI Coprocessor Configuration panel showing:
  - Crypto (0) and (1) Status = deconfigured.
  - The PCI crypto work area, showing both PCICC CHPIDs are “standby/shared/stopped”.

3. Install the PCICA card in service mode.

At completion, you will see:

- The PCI Coprocessor Management panel, showing CHPIDs range 10-13 installed and associated to AP0-AP3.
- The PCI Coprocessor Configuration panel, showing:
  - Crypto (0), (1), (2) and (3) Status = deconfigured.
  - The PCI crypto work area showing both PCICC and both PCICA are “standby/shared/stopped”.

4. Configure On all four CHPIDs.

At completion, you will see:

- PCI Coprocessor Configuration panel, showing:
  - Crypto (0), (1), (2) and (3) Status = configured and PCICC serial numbers are now available.
- PCI crypto work area showing:
  - PCICC/PCICA online/shared/operating)
  - ICSF showing PCICC status “online” and PCICA status “active”.

5. From ICSF, generate symmetric and asymmetric Master Keys for PCICC using pass phrase initialization.

The status of the PCICC coprocessors will now be “active”.

**Note:** During PCICC/PCICA installation, we did not perform a system activation or an image deactivate/activate.

### ***PCICC/PCICA installation steps - walkthrough***

Figure 3-2 shows the CPC Configuration task on the Primary Support Element.

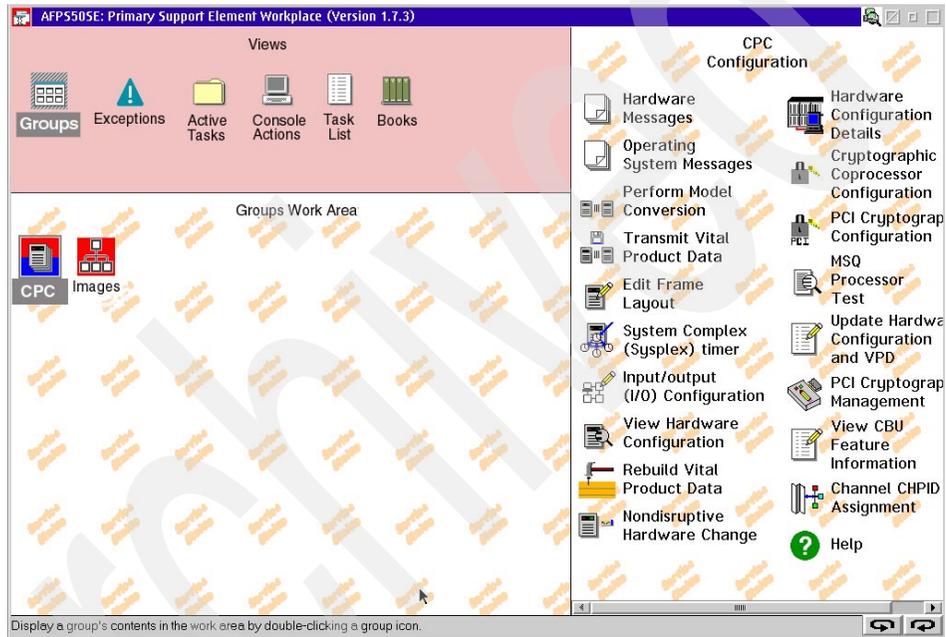


Figure 3-2 CPC Configuration task

From this task, we used the following icons:

- ▶ Cryptographic Coprocessor Configuration for CCF
- ▶ PCI Cryptographic Configuration for PCICC, PCICA and UDX for PCICC.
- ▶ PCI Cryptographic Management for PCICC and PCICA
- ▶ Nondisruptive Hardware Change for installation/removal of PCICC and PCICA cards.

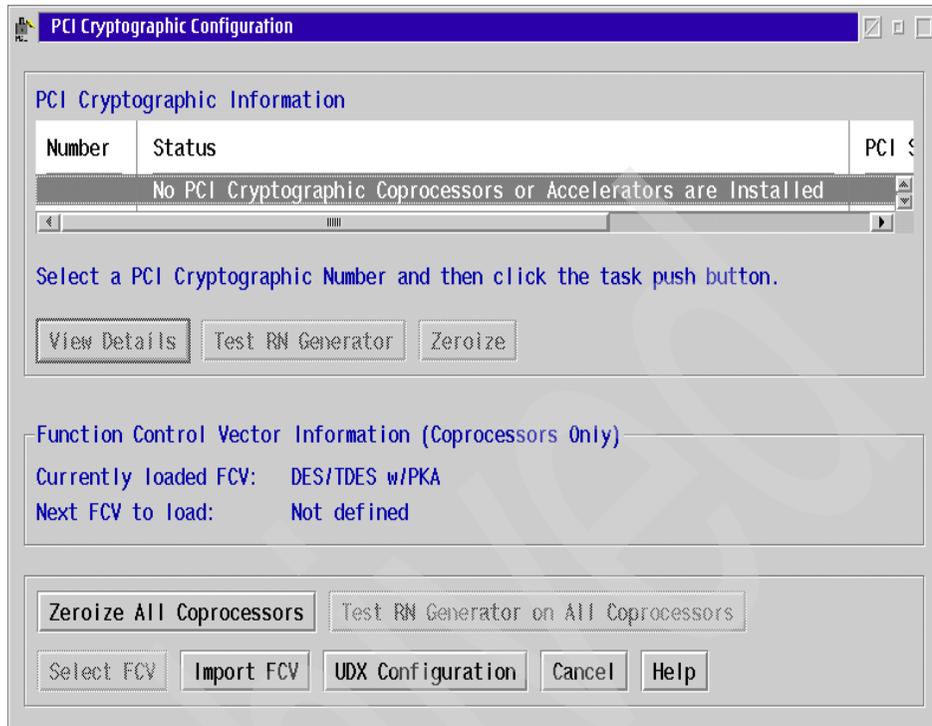


Figure 3-3 PCICC information status: no PCI crypto installed

Figure 3-3 shows the PCI Cryptographic Configuration panel with no PCI crypto installed.

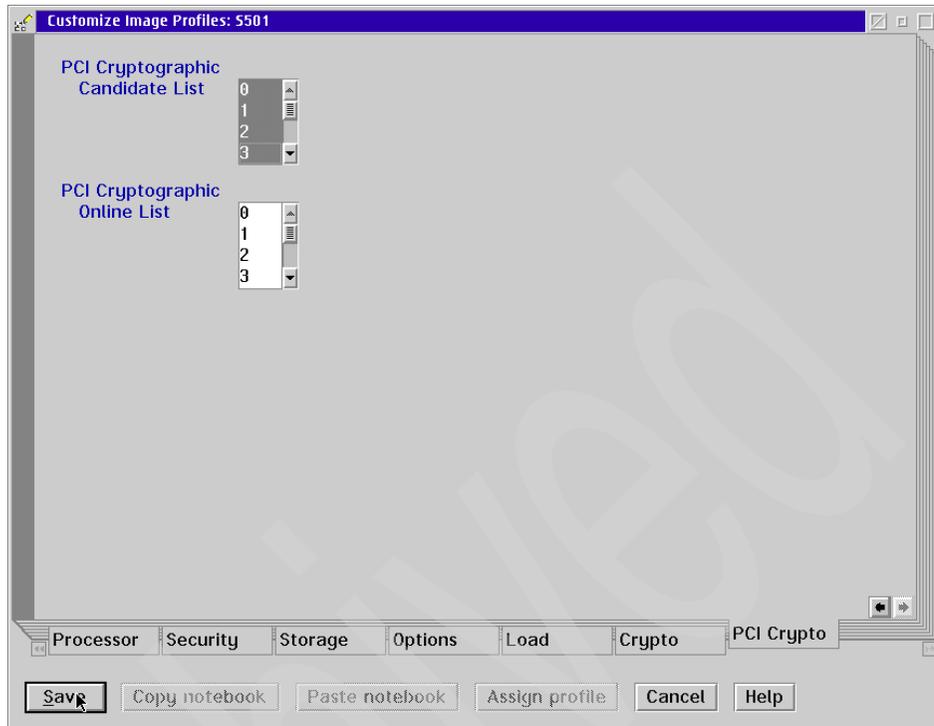


Figure 3-4 Customize image profile - PCI Crypto page

Figure 3-4 illustrates the S501 image profile PCI Crypto page customized with four PCI cryptos (0-3) in the CANDIDATE list and no PCI crypto in the ONLINE list.

After image activation, this setting will allow concurrent PCI crypto card installation, configuration, and activation.

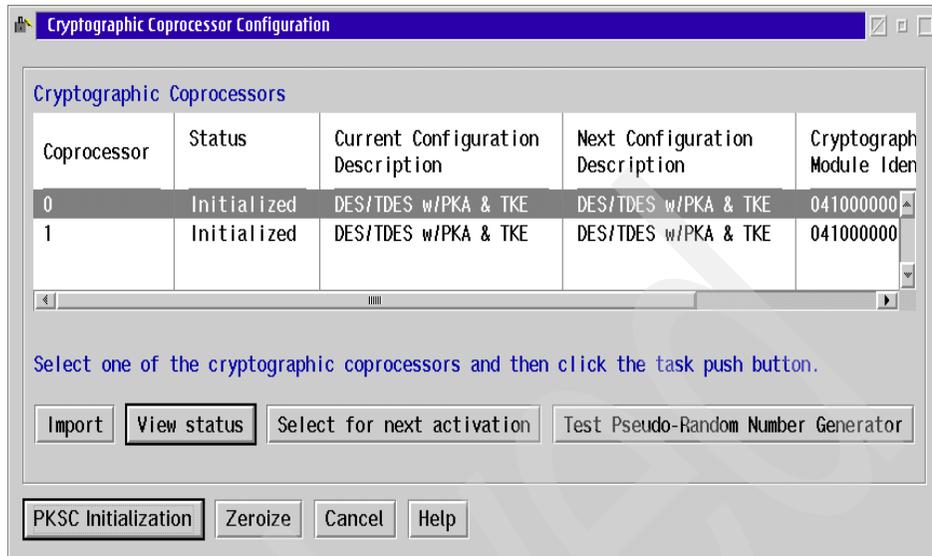


Figure 3-5 Cryptographic Coprocessor Configuration: CCF configured, ICSF up

Figure 3-5 displays the Cryptographic Coprocessor Configuration after both CCF cryptos have been enabled, a POR has been done, and ICSF is up and running.

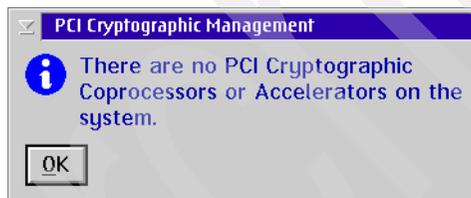
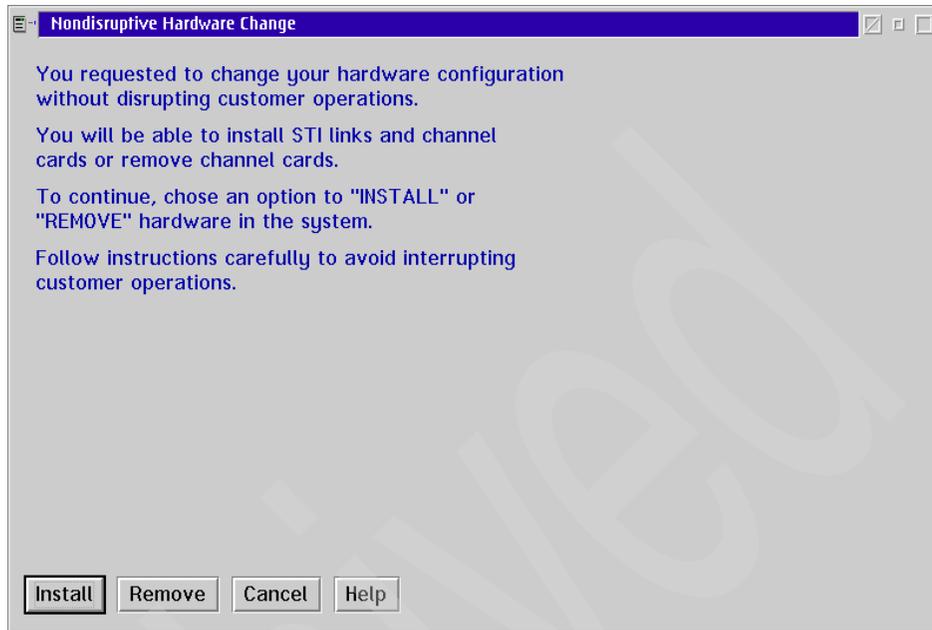


Figure 3-6 No PCI crypto card installed

Figure 3-6 displays the informational message that pops up when you select the PCI Cryptographic Management icon when there is no PCI crypto card installed on the system.

## PCICC installation



*Figure 3-7 Nondisruptive Hardware change selected*

Figure 3-7 shows the first panel displayed after the Nondisruptive Hardware Change option has been selected.

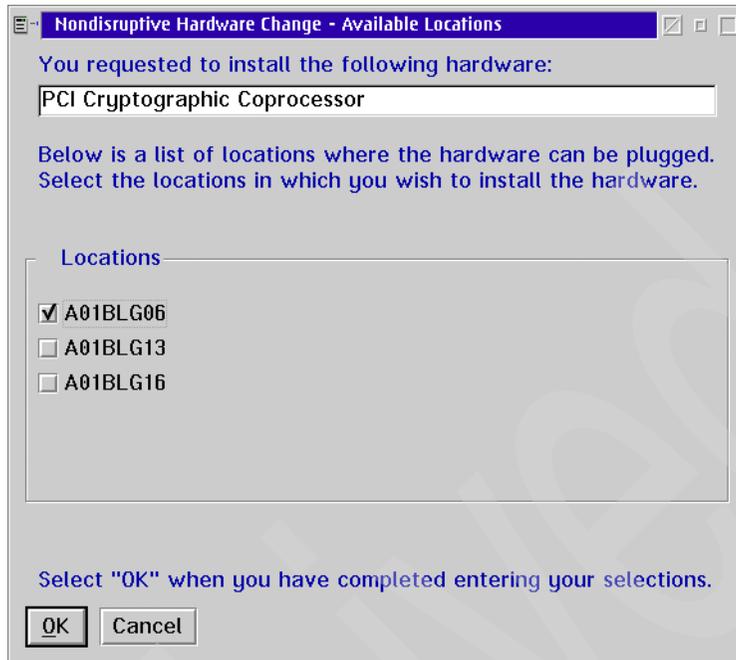
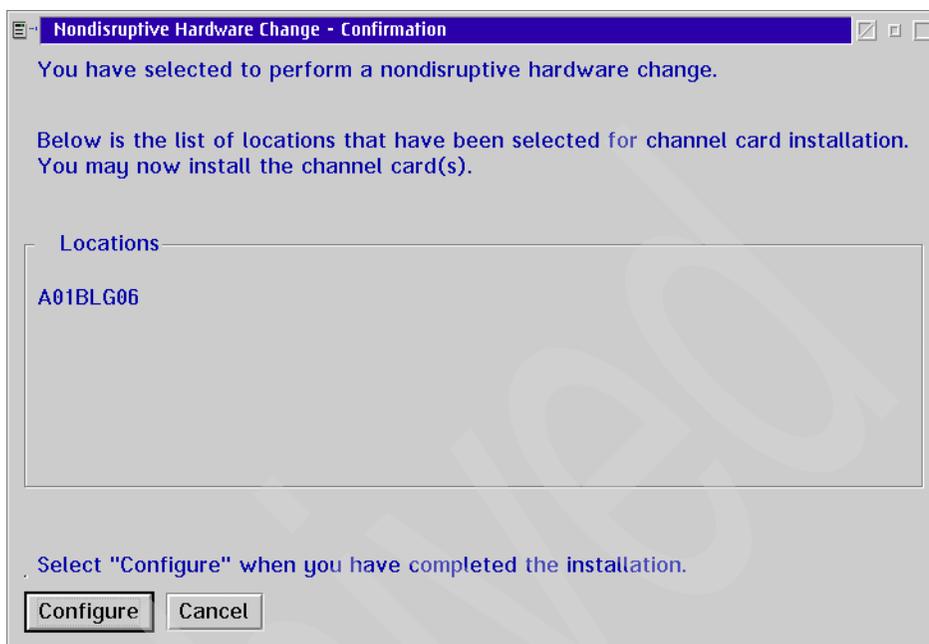


Figure 3-8 Nondisruptive PCICC hardware installation at location A01BLG06

Figure 3-8 shows where to select the location in which you wish to install the PCICC.



*Figure 3-9 Install the PCICC card - then press Configure*

For this operation, an IBM CE would install the PCICC card, and then press **Configure** (see Figure 3-9).

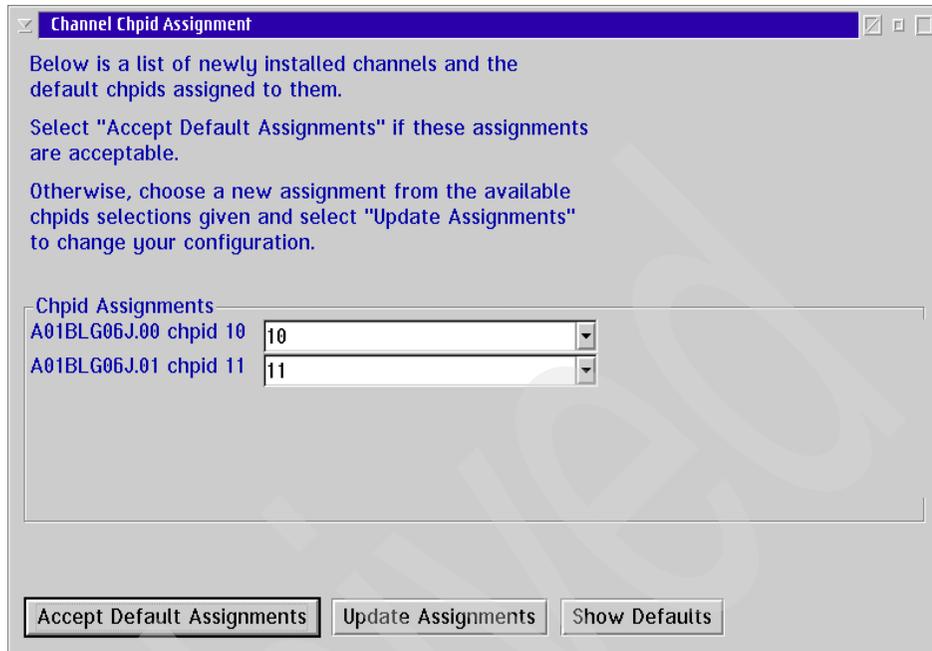


Figure 3-10 PCICC CHPIDs assignments

Figure 3-10 shows the CHPID assignment screen. We recommend that you use the Accept Default Assignments option shown.

**Note:** You can change CHPID mapping later on by using the Change CHPID Management icon on the Support Element.

However, if the number of CHPIDs exceeds 256, including the internal coupling links (IC and ICP), then it will be necessary to remove some CHPIDs before installing the PCICC or PCICA cards.

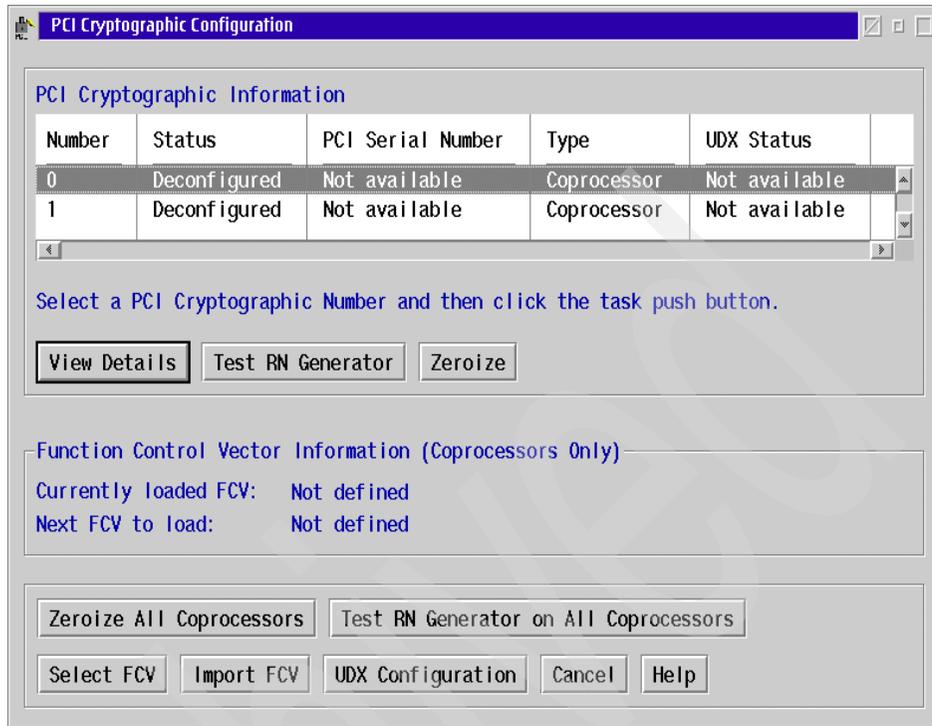


Figure 3-11 PCICC status after hardware installation and before FCV import/select

Figure 3-11 shows the PCICC status after hardware installation and before FCV import/select.

## PCICC enablement (FCV)

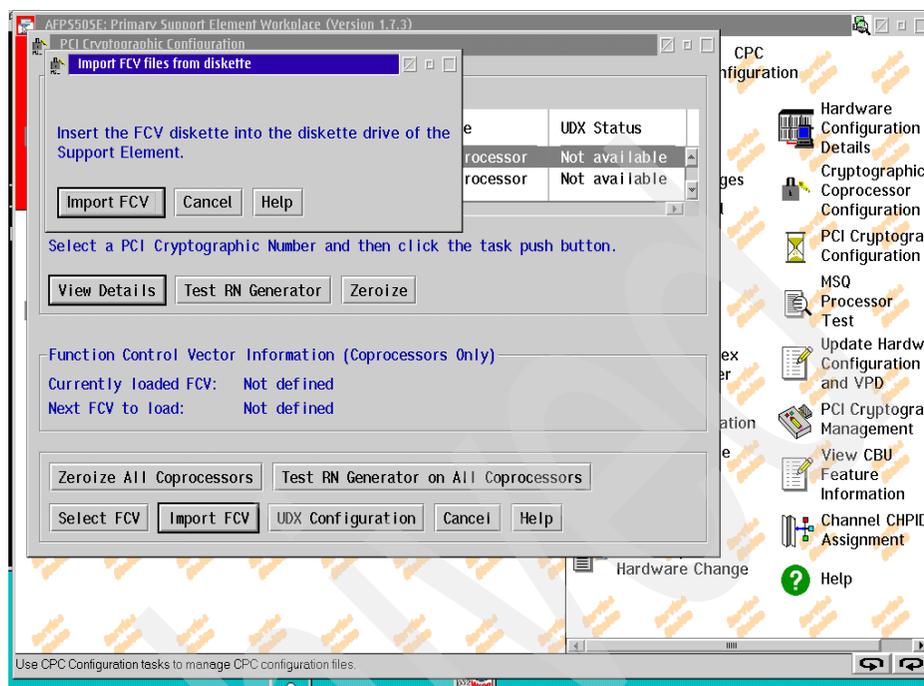


Figure 3-12 IMPORT PCICC FCV from diskette

Figure 3-12 illustrates importing the PCICC FCV from the SE diskette drive. Note that this needs to be done for the *first* PCICC only.

As shown in Figure 3-13, we selected FCV “DES/TDES w/PKA”.

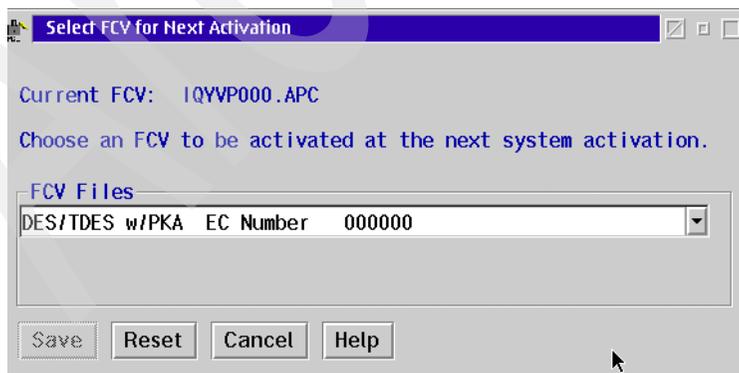


Figure 3-13 PCICC: select FCV file for next activation

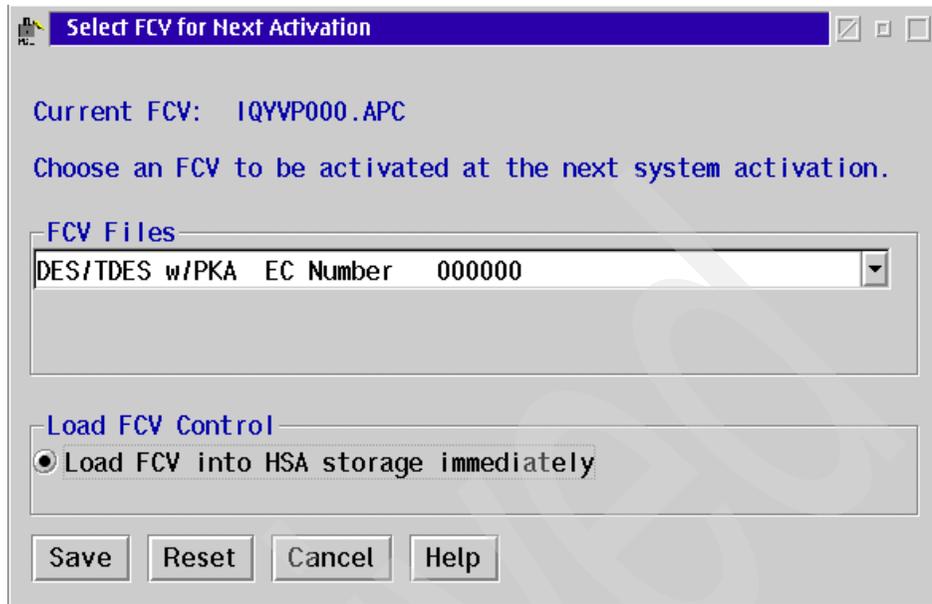


Figure 3-14 Select FCV for next activation (first PCICC only)

For the *first* PCICC, you must use the Load FCV to HSA Immediately option to upload the FCV to HSA, as shown in Figure 3-14.

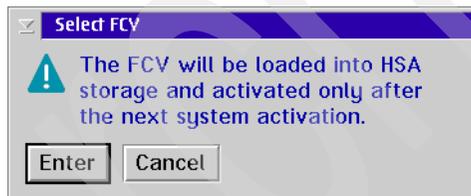


Figure 3-15 PCICC: selected FCV loaded to HSA (first PCICC only)

For the second PCICC, just select FCV and save (the FCV is already in HSA and will be downloaded to PCICC when the PCI crypto card is configured online).



Figure 3-16 PCI Cryptographic Details - one PCI deconfigured

Figure 3-16 shows the PCI Cryptographic Details panel with PCI Crypto (1) deconfigured.

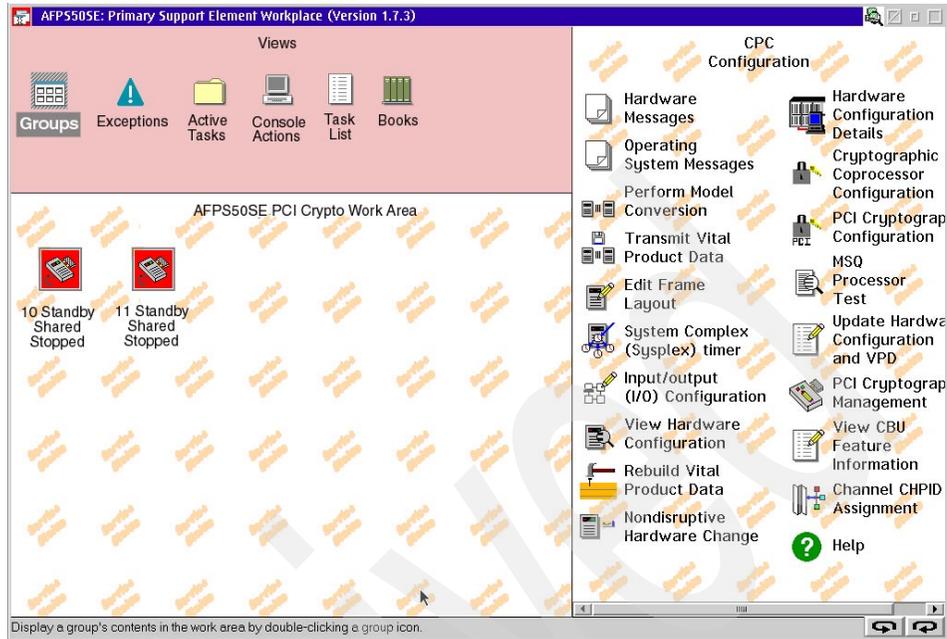


Figure 3-17 PCI Crypto Work Area: PCICC crypto standby/stopped state

Figure 3-17 shows the PCI Crypto Work Area with our two PCI cryptos in standby/stopped state.

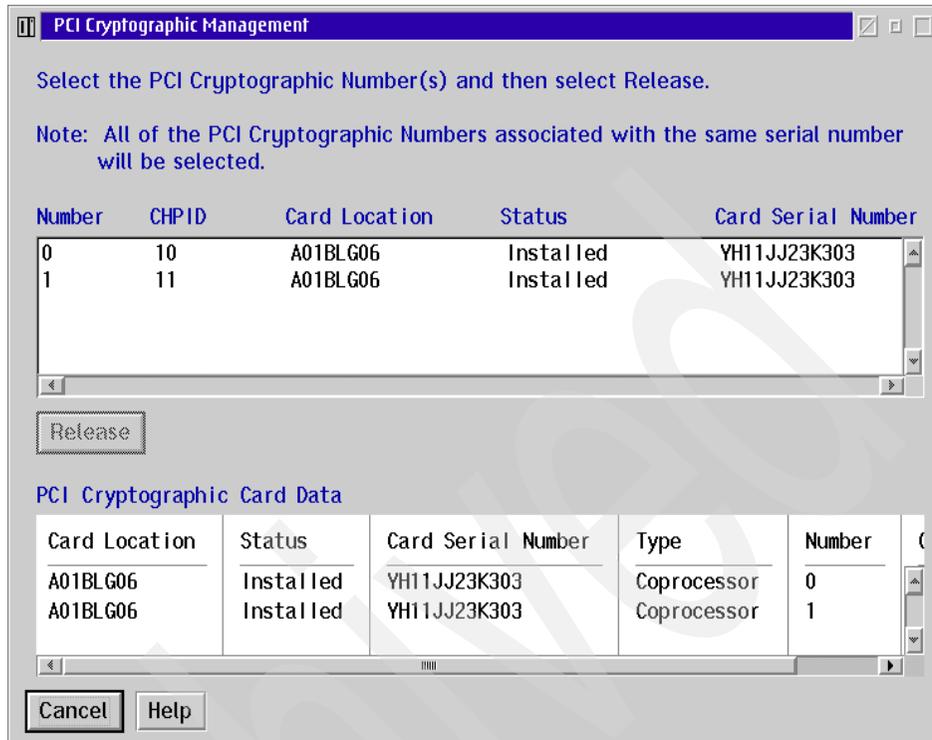


Figure 3-18 PCI Cryptographic Management after PCICC installation

Figure 3-18 shows the PCI Cryptographic Management panel after PCICC installation. This panel shows the relationship between the AP number, CHPID number, the card (book) location, and the card (book) serial number.

## PCICA installation



Figure 3-19 Nondisruptive Hardware: select PCICA channel type

Figure 3-19 shows how to start a nondisruptive hardware installation for a PCICA card.



*Figure 3-20 Nondisruptive Hardware Install: select PCICA location*

As shown in Figure 3-20, the PCICA card will be installed in location A01BLG13.

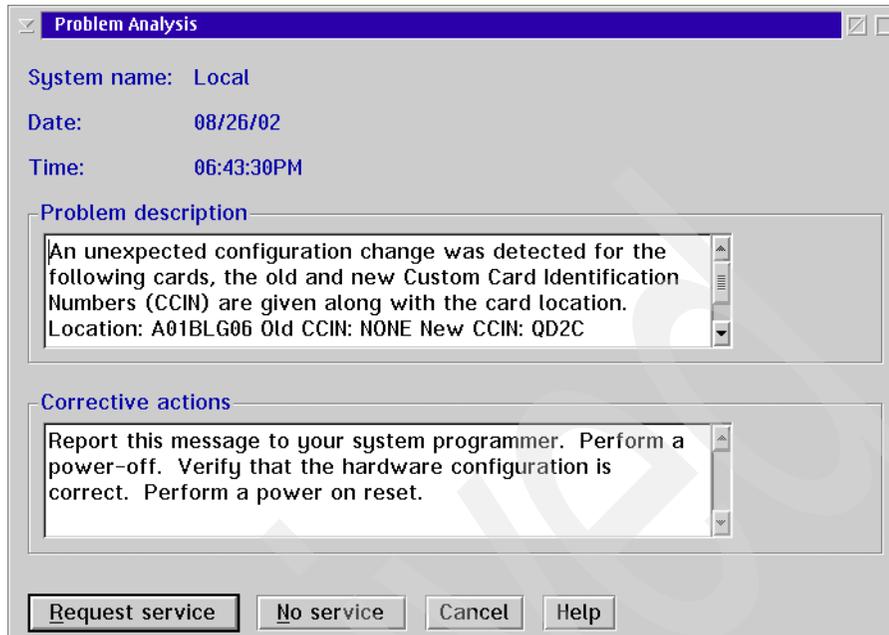


Figure 3-21 Problem Analysis: Unexpected configuration change (hardware message)

After card installation, the hardware message icon will be flashing, indicating that an unexpected configuration change was detected for the given card; see Figure 3-21.

No service is needed, and the message can be deleted.

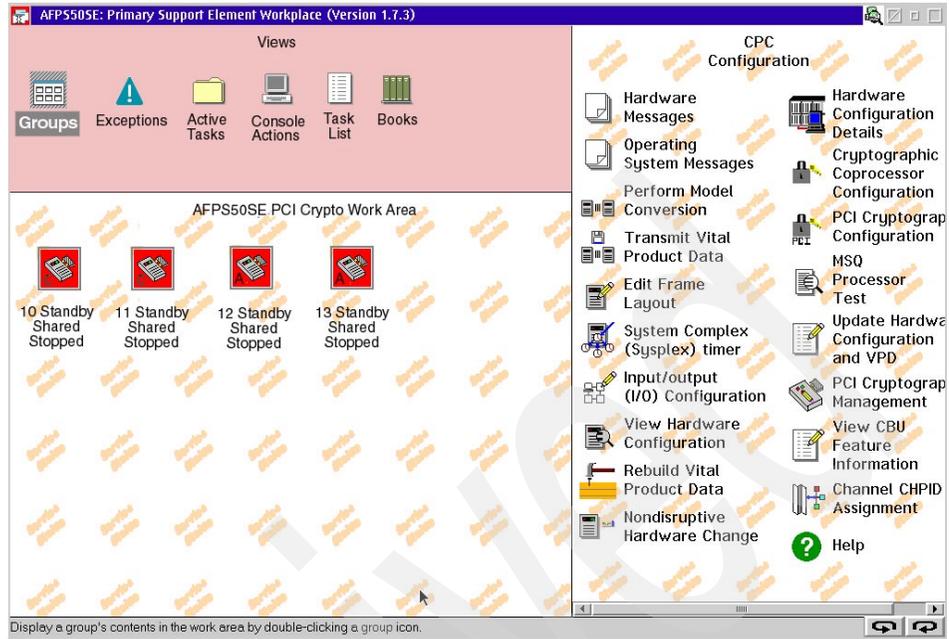


Figure 3-22 PCI Crypto Work Area: PCICC/PCICA in Standby/Stopped status

Figure 3-22 represents a view from the PCI Crypto Work Area with all four PCI crypto CHPIDs in Standby/Shared/Stopped status.

The next step is to vary them online. This can be done only from the Support Element (CHPID Operations task); refer to Figure 3-23 on page 78.

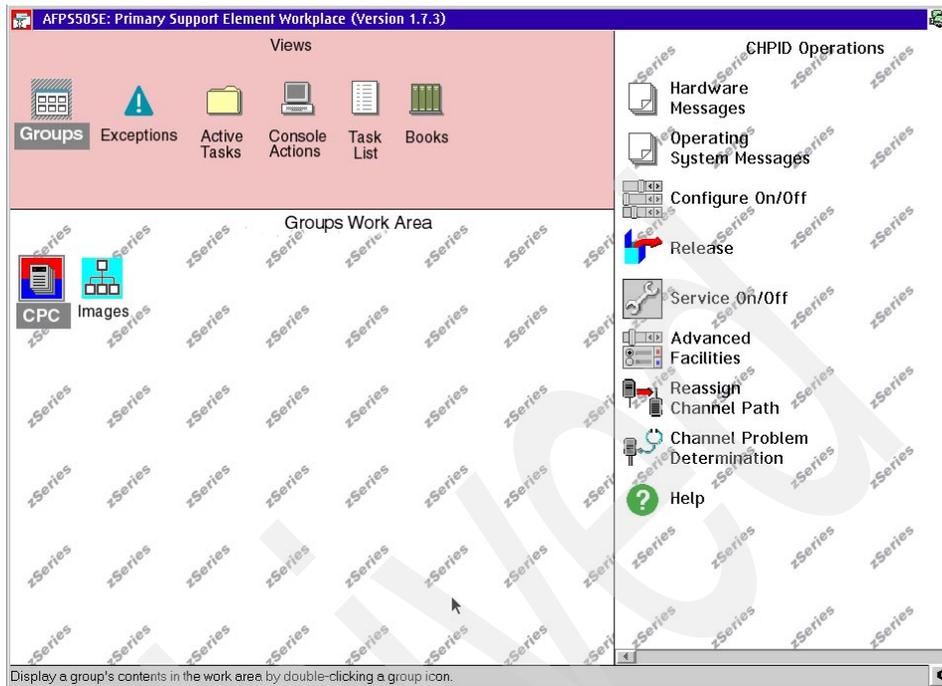


Figure 3-23 CHPID Operations task

The PCI-crypto CHPID must be configured online at the Support Element. This will trigger the loading of the microcode and FCV into the PCICC, and make it available for crypto-enabled LPARs.

From the PCI work area, select all PCI cryptos and drag and drop them to Configure On/Off. Toggle them On, and then Apply.

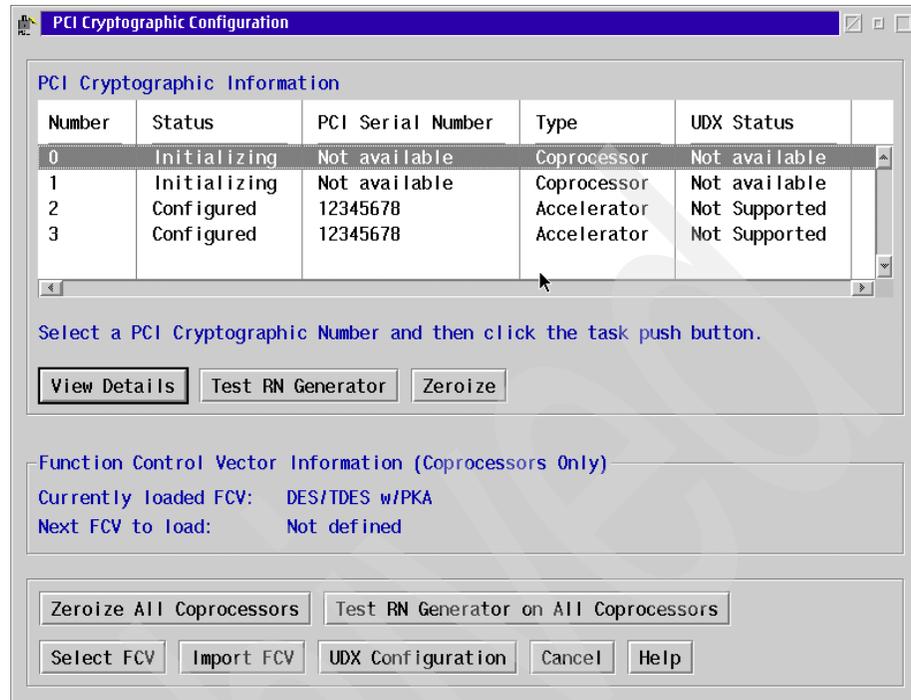


Figure 3-24 PCI Crypto Configuration: PCICC 0/1 initializing

After Configure On has been applied to all PCI cryptos, the status changes as follows:

- ▶ PCICC
  - “Deconfigured” to “initializing”, and then to “configured”
- ▶ PCICA
  - “Deconfigured” to “configured”

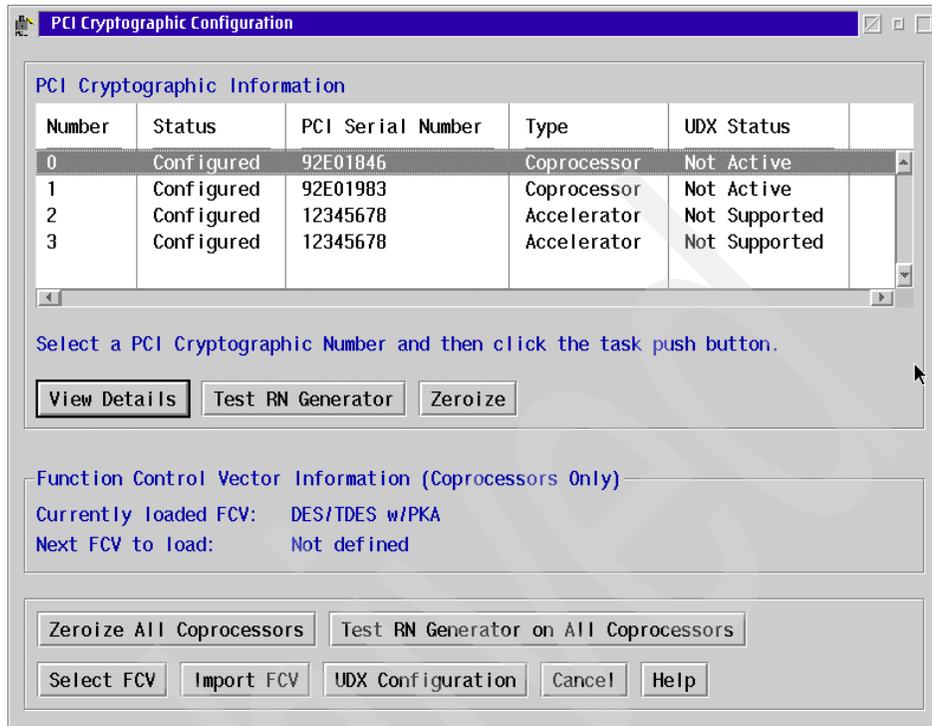


Figure 3-25 PCI Cryptographic Configuration: all PCIC cryptos configured

The PCI Serial Number information is related to each individual crypto card, and it appears on the PCI Coprocessor Configuration panel after the PCICC card has been configured.

**Note:** This PCI Serial Number information is different from the Card serial number, which is related to the book.

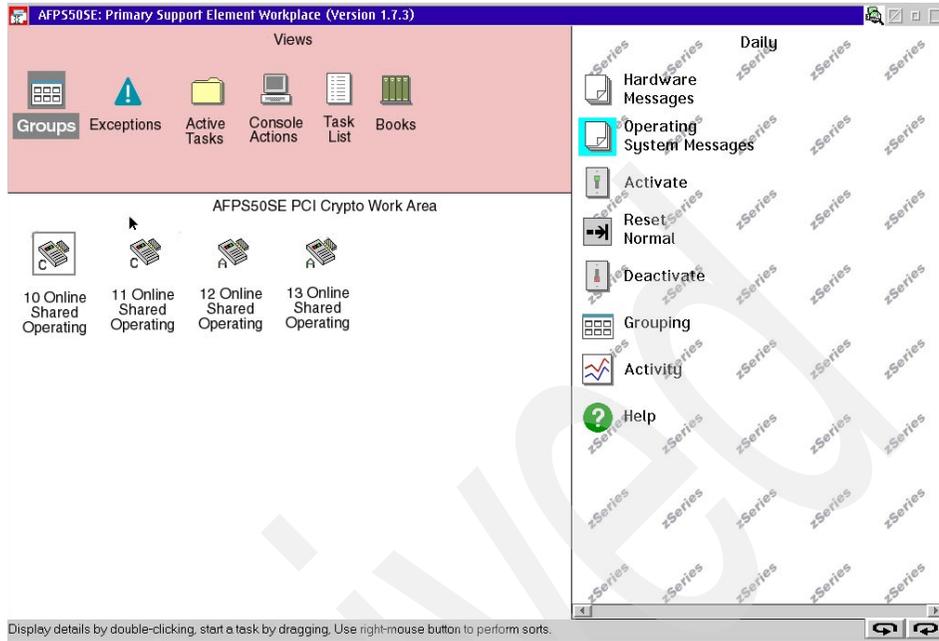


Figure 3-26 PCI Crypto Work Area status: Online/Shared/Operating

Figure 3-26 illustrates the status of each PCI crypto CHPID by means of an icon. The icon is labelled C for crypto coprocessor, and A for crypto accelerator.

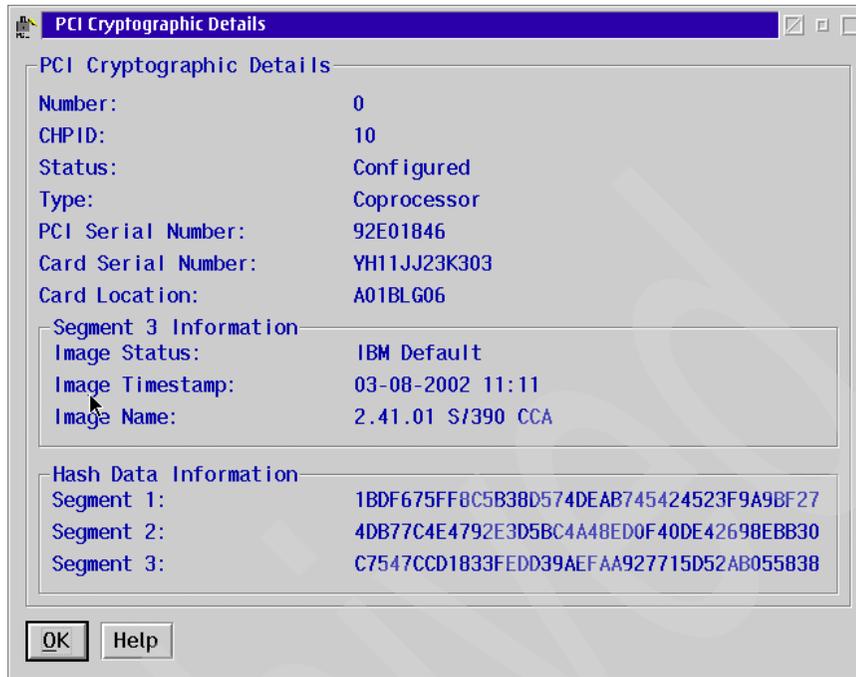


Figure 3-27 PCI Cryptographic Details: PCICC crypto (0) configured

Figure 3-27 shows the PCI Cryptographic Details panel with PCICC crypto (0) configured, as well as its PCI serial number, segment 3 information, and segment 1 to 3 hash data information.



Figure 3-28 PCI Cryptographic Details: PCICA crypto (2) configured

Figure 3-28 shows the PCI Cryptographic Details panel after crypto (2) was selected. This is a PCI coprocessor accelerator.

### 3.3.2 Second scenario (adding PCICC concurrently)

Following are the major steps in this process.

1. Change the S501 image profile to define PCI cryptos in the candidate list as follows:
  - Candidate list: AP0 to 15
  - Online list: AP2 and AP3 (PCICA)
2. Deactivate the S501 image.
  - CHPIDs 10,11 are standby/shared/stopped.
  - CHPIDs 12,13 are online/shared/operating.
3. Configure CHPIDs 10 and 11 online.
  - CHPIDs 10,11,12, and 13 are online/shared/operating.
  - The PCI Coprocessor Configuration menu shows all crypto coprocessors with a status of “Configured”.

4. From ICSF, PCICC is in “Online” status, while PCICA is in “Active status”.
5. From ICSF, generate Master Keys for PCICC. At completion, the PCICC is in “Active” status.

### 3.3.3 Third scenario (UDX installation - hardware side)

The UDX Package file is a binary file named IQYVPxxx.UDX, which comes on a diskette. It is possible to import and activate a UDX from either the Support Element or the Hardware Management Console on the GA2 level of z900. Note that importing the UDX file from the HMC is possible, but the UDX file will be transferred using a non-secure connection.

(Refer to Chapter 4, “Installation, configuration and startup of ICSF” on page 101 for details about installation on the host side.)

The following procedure is taken from the *zSeries Support Element Operations Guide*, SC28-6813. (In the section that follows, we illustrate these steps and walk you through the details, using Figure 3-29 on page 85 through Figure 3-39 on page 90.)

1. Log on to the Support Element in system programmer or service representative mode.
2. The base Cryptographic Coprocessor feature must be enabled, the PCI Cryptographic Coprocessor feature must be installed, and the CPC must be powered on.
3. Drag and drop the CPC on the PCI Cryptographic Configuration task to start it.
4. Select the **UDX Configuration** push button to configure the PCI Cryptographic Coprocessors for UDX. The UDX Configuration window displays detailed information for the PCI Cryptographic Coprocessor configured for UDX capability, and provides push buttons for working with them.
5. Select the **Import** push button to import the UDX file from the diskette to the Support Element hard drive. The Import window displays.  
Insert the UDX diskette into the diskette drive.
6. Select the **Activate** push button to load the UDX data to the PCI Cryptographic Coprocessor. Use the online help for more information on UDX configuration.

**Important:** The UDX activation or Reset to IBM default actions will write over the Segment 3 code on the selected PCI crypto coprocessor. As a consequence, the Master Keys and the configuration data previously entered from the TKE Workstation will be lost (it has the same effect as replacing a PCI crypto card).

Therefore, after the new code is activated:

- ▶ The customer will need to restore PCICC Master Keys for all domains (Pass Phrase Initialization from ICSF).
- ▶ If the customer is using a TKE Workstation, then the corresponding configuration data for the PCICC needs to be restored from the TKE for all domains; refer to Chapter 5, “Customizing PCICC and CCF using TKE V3.1” on page 147 for more information.

## UDX installation - walkthrough

Figure 3-29 shows the UDX configuration panel with the import function. No UDX file imported yet. Segment 3 contains IBM default code.

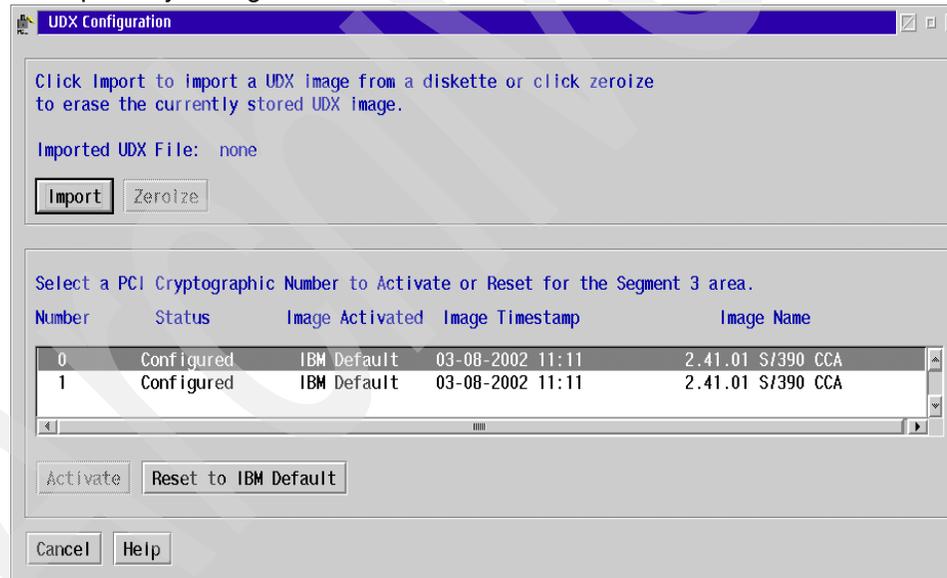


Figure 3-29 UDX Configuration: Import selection

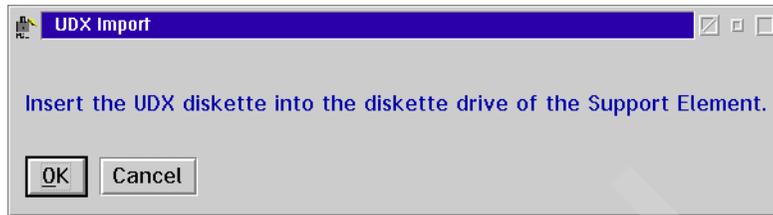


Figure 3-30 UDX Import: insert UDX diskette

Figure 3-30 shows the insert UDX diskette display.

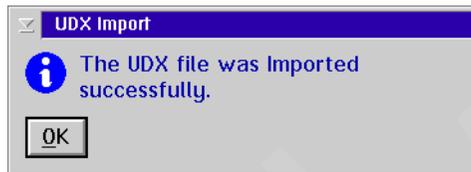


Figure 3-31 UDX file imported successfully

Figure 3-31 shows the import was successful, and Figure 3-32 shows the UDX file is ready for activation.

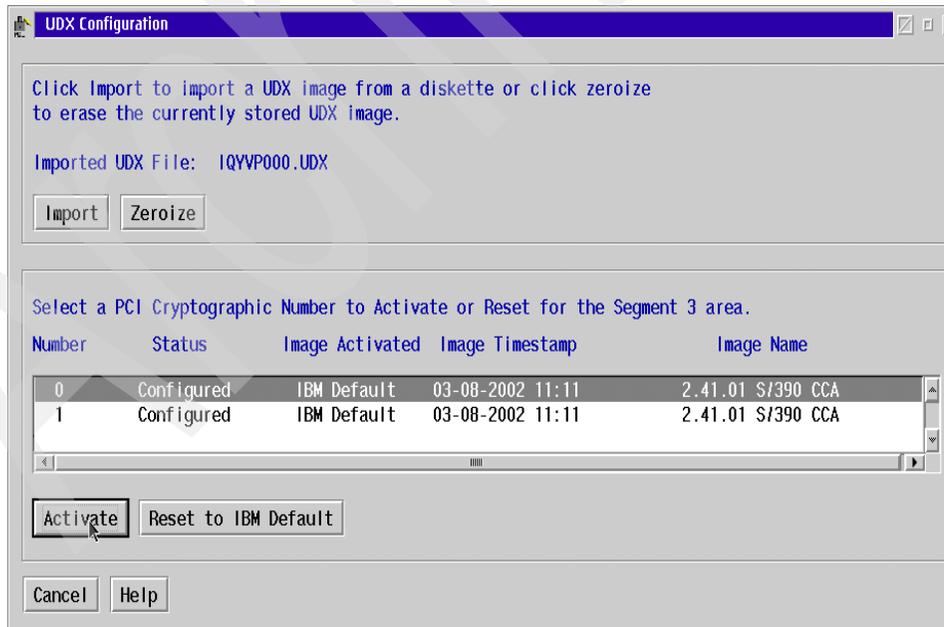


Figure 3-32 UDX Configuration: UDX file imported, ready for Activation

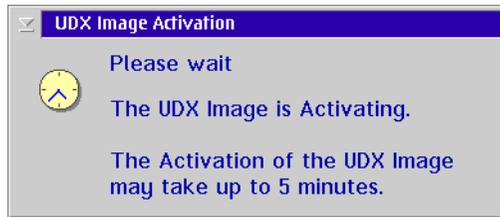


Figure 3-33 UDX Image is activating on selected PCICC

Figure 3-33 shows the activating message.



Figure 3-34 UDX activation complete

Figure 3-34 shows that the activation is now complete.

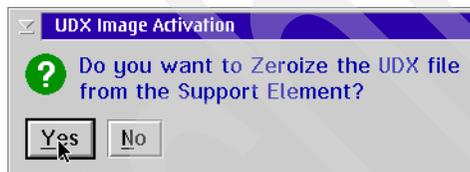


Figure 3-35 UDX Zeroize option

Figure 3-35 shows the Zeroize option, which is used to erase the UDX file from the SE hard disk. This selection panel pops up at the end of UDX activation.

Figure 3-36 shows a Warning panel that pops up after the Zeroize option is selected from the UDX configuration window. Pressing Enter will erase the UDX files from SE hard disk; this is a security option.

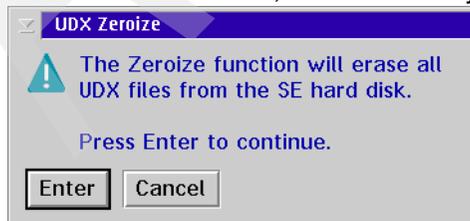


Figure 3-36 UDX Zeroize: Zeroize selection from Import panel

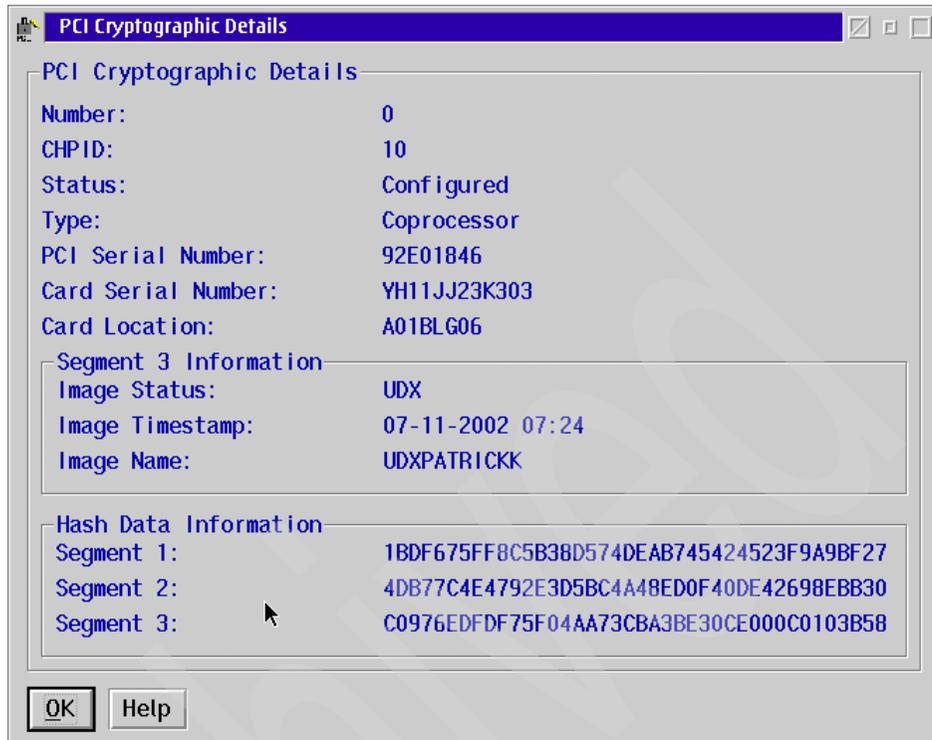


Figure 3-37 PCI Cryptographic Details: PCI crypto (0) after UDX activated

Figure 3-37 shows the PCI Cryptographic details for PCI crypto (0). Note the following changes after UDX activation.

- ▶ System 3 image information:
  - status = UDX
  - new image name
- ▶ Hash data information: Segment 3 has changed

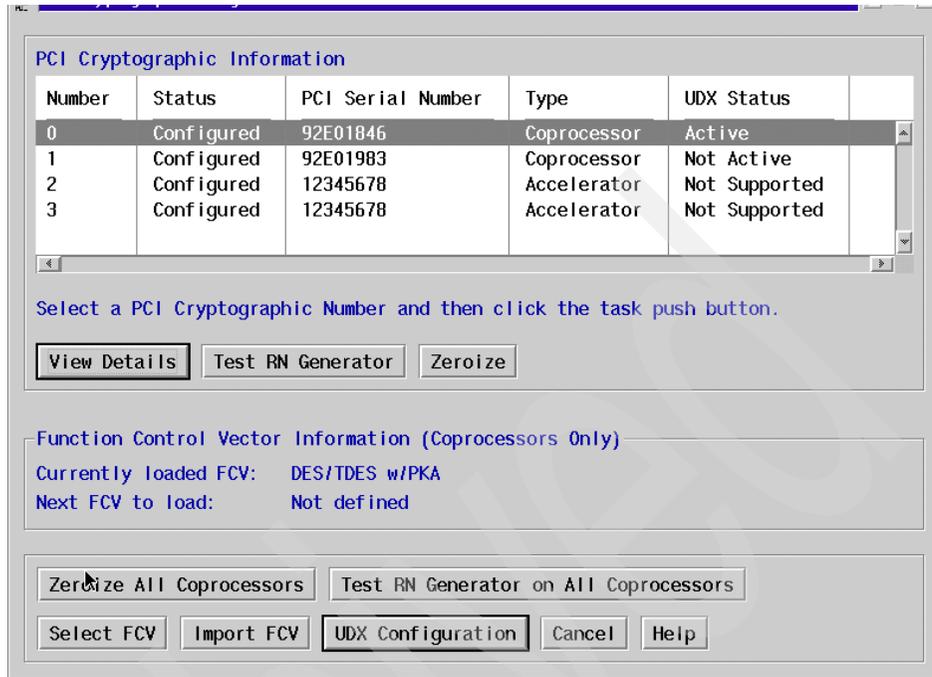


Figure 3-38 PCI Cryptographic Configuration: UDX activated on PCI Crypto (0)

Figure 3-38 shows PCI crypto (0) configured after UDX activation and PCI crypto (1) configured with the IBM default code.

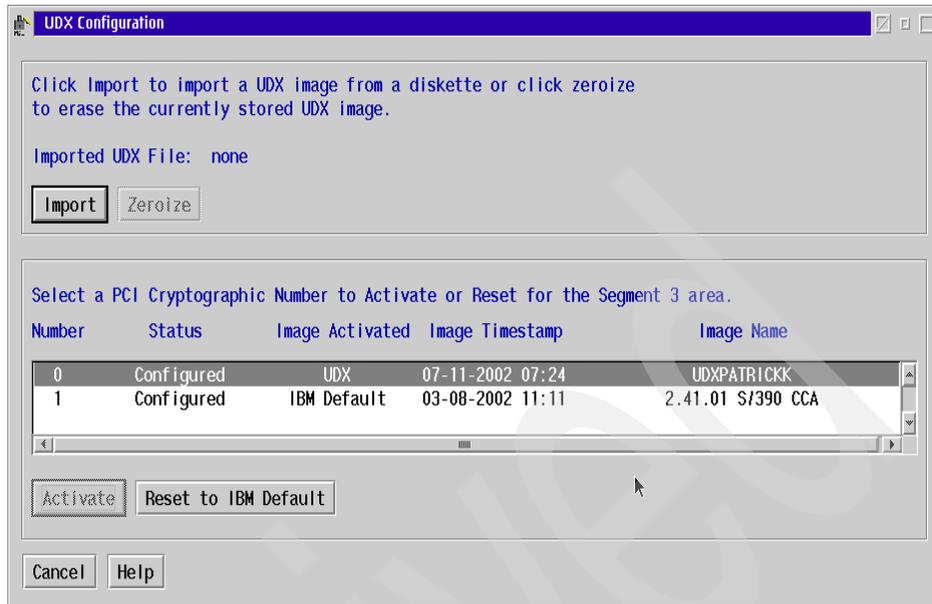


Figure 3-39 UDX Configuration panel: PCI crypto (0) configured

Figure 3-39 shows PCI crypto (0) configured with the UDX image name of UDXPATRICKK and PCI crypto (1) configured with the IBM default code.

From ICSF, first generate Master Keys, then activate PCI crypto (0). From the TKE Workstation, recognize the PCICC module and then redefine “Authorities” /“Roles” and send updates to the PCICC.

Note that there is a case where the UDX file activation does not erase the security data so Master Keys and TKE configuration data for the PCI crypto are still valid.

### 3.3.4 Removing one PCICC

Remove a PCICC card from the system by executing the following steps:

1. Deactivate AP0 and AP1 in the ICSF PCICC Management panel.
2. Configure CHPIDs 10 and 11 offline at the Support Element.
3. Put CHPIDs 10 and 11 in Service On (CHPID task).
4. Invoke the Nondisruptive Hardware Change option at the Support Element, and remove the PCICC Card at position LG06.
5. If needed, release the two APs, using the PCI Cryptographic Management icon.

## 3.4 The z900 channel subsystem

In the following sections, we briefly describe the z900 internal structure and the channel plugging rules.

For more details, refer to the IBM Redbook *IBM eServer™ zSeries 900 Technical Guide*, SG24-5975.

### 3.4.1 The z900 internal structure

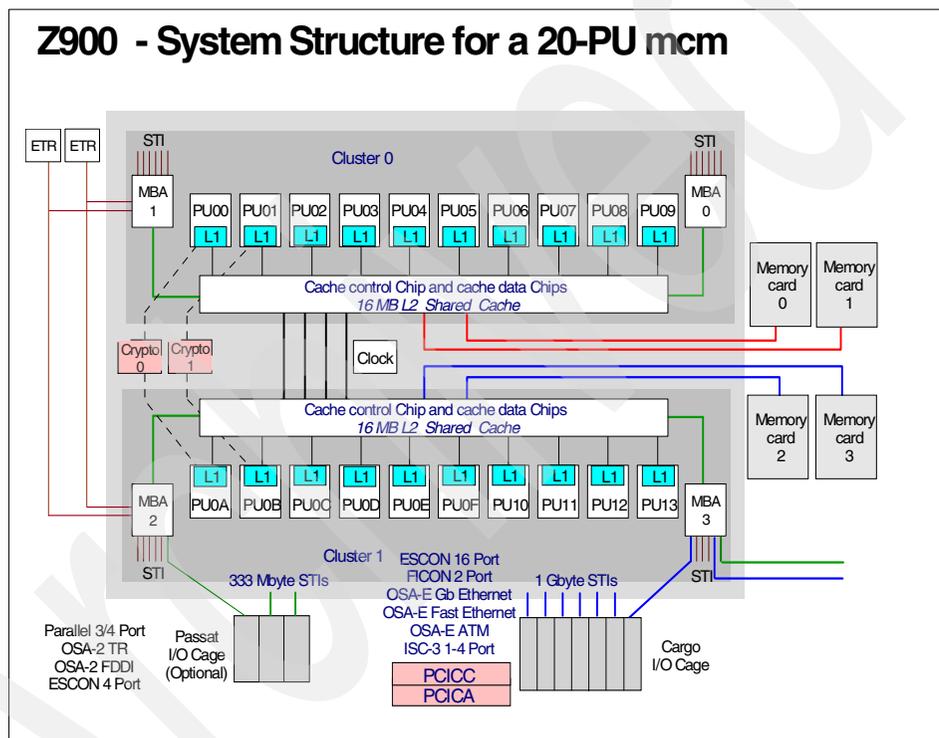


Figure 3-40 Z900 internal system structure

The central processor module (MCM) contains the z900 bi-nodal Processor Unit (PU) clusters. All z900 Server models have two clusters of PUs. Each of them are supported by Storage Control chips, Storage Data chips, Memory Bus Adapter (MBA) chips, and the clock chip.

The Channel subsystem (CSS) controls all the system Input/Output operations.

The CSS of the z900 general purpose models is able to execute three types of I/O operations:

- ▶ Channel Command words (CCWs)-based I/O operations.
- ▶ Queued Direct I/O (QDIO) operations.
- ▶ Messages operations.

An I/O operation is always initiated by a Central Processor (CP). The CP then transfers control to one of the System Assist Processors (SAPs), which is a processing unit (PU) running specialized microcode. The SAP then selects the required channel to execute the operation and is responsible for path rotation to optimize access to multi-path Control Units like Direct Access Storage Devices (DASDs) and tapes.

The selected channel is then instructed to perform the I/O operation. Data transfer either takes place from Main Storage to I/O (write operation) or from I/O to Main Storage (read operation). When the I/O operation is completed at the device level, the SAP is solicited to present ending status to the CP on behalf of the channel.

All control communications between SAP and channels, and all data exchanges between Main Storage and channels, use the following z900 CSS hardware:

- ▶ Memory Bus Adapter (MBA)
- ▶ MBA Self-Timed Interface (STI) ports
- ▶ Self-Timed Interface (STIs) cables
- ▶ For z900 I/O cage (FC 2023):
  - Self-Timed Interface - Multiplexer (STI-M) card
  - I/O cage internal wiring
  - Channel cards

PCICC and PCICA are two separate cards with separate functions. PCICC and PCICA can be on the same STI because they serve different functions, but additional PCICC or PCICA cards should be spread across STIs.

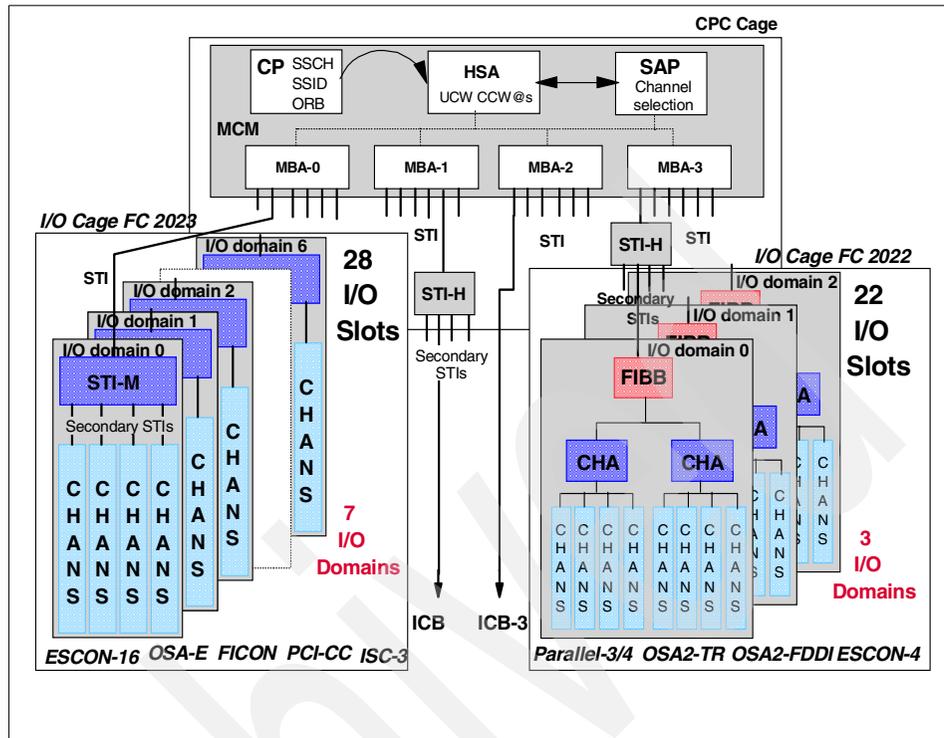


Figure 3-41 z900 I/O cages and channel number topology

Only z900 I/O cage FC2023 can contain PCICC and PCICA cards; see Figure 3-41.

### 3.4.2 View Hardware Configuration icon (CPC configuration task)

You can use the View Hardware Configuration icon to obtain information about configuration details; see Figure 3-42 on page 94.

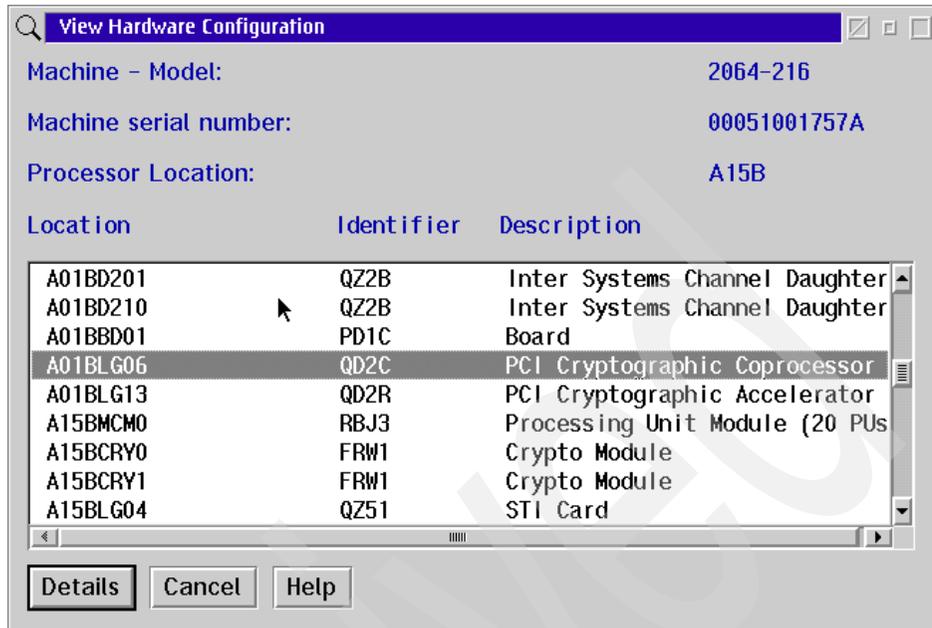


Figure 3-42 View Hardware Configuration panel

Figure 3-43 displays details about the PCICC card.

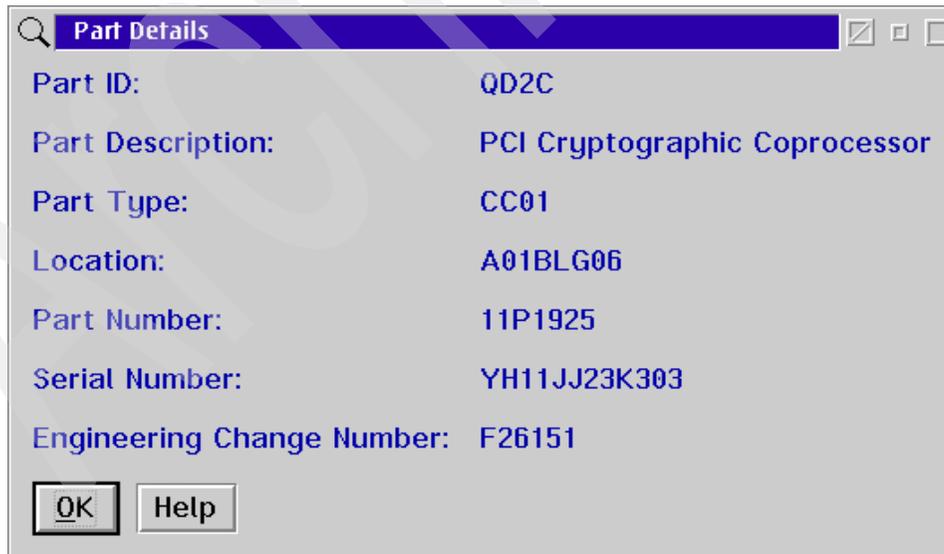


Figure 3-43 View Hardware Configuration Details: PCICC card

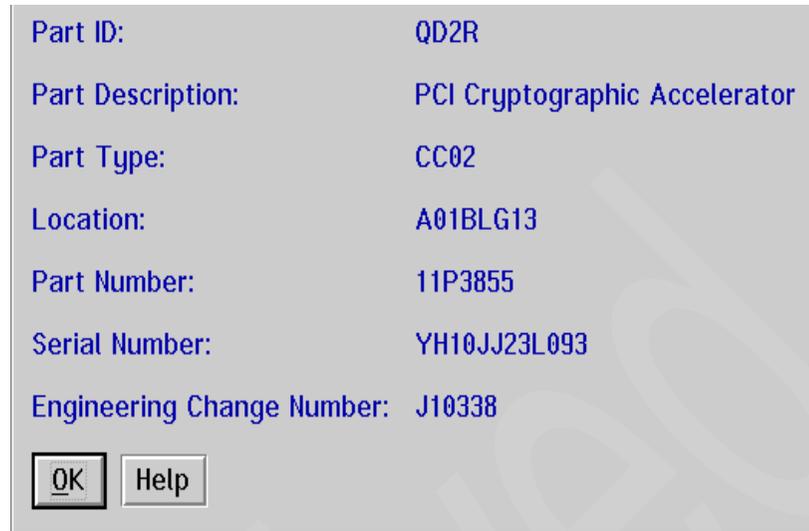


Figure 3-44 View Hardware Configuration Details: PCICA card

Figure 3-44 displays details about the PCICA card.



Figure 3-45 View Hardware Configuration Details: CCF module 0

Figure 3-45 displays details about CCF module 0.

## 3.5 Planning list items

In the following sections we discuss planning and installation considerations and requirements.

### 3.5.1 Capacity planning considerations

Refer to Appendix D, “Crypto performance considerations” on page 325 for detailed capacity planning and performance considerations.

### 3.5.2 Installation of the ordered PCICCs

Note the following installation considerations.

#### On new build machines

- ▶ All PCI crypto cards (FC 0861 PCICC) are shipped in thermal containers to avoid the risk of thermal tamper detection during shipment.
- ▶ The PCICC cards are removed from the machine after testing at the system manufacturing plant, and the Customer Engineer has to replug the PCICC cards at the customer's site according to the instructions given in the installation manual.
- ▶ The FCV crypto enablement diskette is also imported during the installation, once per system only, even if more than one feature code 0861 is installed.
- ▶ If the TKE has been ordered with feature code 0869/0879 or 0866/ 0876, then the IBM-4758 adapter is removed from the workstation at the plant of manufacture and shipped in a thermal container. The Customer Engineer has to replug the 4758 crypto adapter at the customer's site.

#### MES upgrade

For MES upgrades, installation of the PCICC cards can be done concurrent with system operations if *all* of the following conditions are met:

- ▶ At least one CCF is enabled and in operation, with an enablement feature matching the PCICC-intended cryptographic configuration.
- ▶ Because MES Installation Instructions are always generated for a specific machine serial number, it is very important that the last Vital Product Data (VPD) transmitted to IBM for this machine be accurate.

#### Miscellaneous recommendations

- ▶ Make sure that any unique customer requirements have been conveyed through the ordering process and taken into consideration prior to starting installation.

- ▶ If you move PCICC cards, those cards will be zeroized. Refer to Chapter 2, “PCICC and PCICA product overview” on page 15 for more information about this subject.
- ▶ The PCICC and PCICA CHPIDs are not defined in the IOCDs, but a CHPID definition conflict may occur if a CHPID is already defined in the IOCDs with the same number as the PCICC. This will prevent I/O dynamic reconfiguration, and if a system power-on Reset is performed, neither the defined CHPID nor the PCICC will come online and a POR error message will be received.

## 3.6 PR/SM setup

This item is discussed in detail in Chapter 4, “Installation, configuration and startup of ICSF” on page 101.

**Notches** PCICC or PCICA online and candidate lists in the partition image profile cannot be updated dynamically.

There are no migration tools provided for migrating to TKE V3.1. You must consider the impact on the items mentioned in the following sections and perform the appropriate actions.

### 3.6.1 Host definitions

The V2 host definitions are for VTAM® APPC-connected hosts. You must redefine these hosts for TKE V3.1 support of TCP/IP only.

### 3.6.2 CCF crypto modules, domains, and authority definitions

The TKE V2.x TKECM dataset is not compatible with the TKE V3.1 data set. You must redefine the CCF crypto modules, domains, and authority in the new TKECM data set. If you plan to use the same dataset name for V3.1 as you used for V2.0, you must *delete* or *rename* the existing data set.

### 3.6.3 Authority signature keys on IBM Personal Security Card (PSC)

Authority signature keys on PSC cannot be used on the TKE V3.1. The following alternatives are suggested:

1. While still working with the TKE 2.x workstation: change the signature requirements so that signature keys stored in PSCs are not needed any more.

From the Crypto Module window, update the appropriate commands with the new signature requirements. Remove any authority whose signature key was stored on PSCs.

**Note:** Be careful, however, not to create security exposures by removing these signature requirements.

2. As an alternative, still on the TKE 2.x Workstation, you can generate and load new signature keys to the host as follows:
  - From the Authority Administration window, generate a new signature key and save it to a binary file. Read the public modulus. Send the updated authority definition the host, copy the binary file to a diskette, and restore the diskette files to the new TKE Workstation.

### 3.6.4 Authority signature key in the TKE Workstation key storage

There is no direct migration for PKA tokens from the 4755 crypto adapter card to the 4758.

### 3.6.5 IMP-PKA keys in the workstation key storage

You must generate and load new IMP-PKA tokens in the TKE V3.1 key storage.

### 3.6.6 Migration of master or operational key parts on PSC

Because TKE V3.1 does not support the 4754 SIU or the PSC, you need to migrate the PSC data blocks to files. You can accomplish this while still working with TKE V2 by using the TSS HIKM utility. *TKE Workstation User's Guide 2000*, SA22-7524, describes this operation in detail in the chapter "Migrating from TKE V2 to TKE V3.1".

## 3.7 Site security policy

The site security policy must take into account the following:

- ▶ The impact of new concepts in the cryptographic services (such as "retained keys"), in terms of the availability of services.
- ▶ The new RACF profiles pertaining to the PCICC-specific panels and functions.

- ▶ The changes brought by the new TKE Workstation, as follows:
  - The communication network is a TCP/IP network. A “listener” host transaction program runs in the host z/OS and can be secured using new RACF profiles.
  - Both the TKE cryptographic adapter and the PCICC management introduce the notion of roles and profiles. These notions are further explained in Chapter 5, “Customizing PCICC and CCF using TKE V3.1” on page 147.

Archived



## Installation, configuration and startup of ICSF

In this chapter we describe an overview of the physical installation of the PCICC and PCICA cards and all the initialization and setup steps to be performed to enable the customer to use the installed PCICC/PCICA cards. We also describe functional changes on ICSF and panels including User Defined Extensions (UDX) management.

We do not describe the TKE Workstation setup, which is covered in Chapter 5, “Customizing PCICC and CCF using TKE V3.1” on page 147.

The first steps of the initialization are performed at the Support Element or Hardware Management Console (HMC). This includes the cryptographic coprocessor assignment and authorization setup in the image profiles of the logical partitions, which must be done before activating the partitions.

- ▶ ICSF is used to enter the Master Keys in the hardware and to initialize and manage the cryptographic keys data sets.
- ▶ The Cryptographic Coprocessor Facility (CCF) must be in operation before using the PCICC cards, and the CCF cryptographic configuration must match the PCICC FCV feature codes.

### Notes:

▶ “Operator command”

Several manuals that we used while writing this redbook refer to using an “operator command” to vary the PCICC/PCICA online or offline.

No such command exists that will allow z/OS operators to configure a PCICC/PCICA online or offline from the system console. Configuring a PCICC/PCICA online or offline can only be accomplished at the system Support Element or HMC.

▶ Terminology - “online”

The term “online”, when applied to the PCICC/PCICA, requires precision regarding the context in which it is used, as follows:

- If referring to the PCICC/PCICA CHPID, the meaning of “online” is the same as for a CHPID being in an online or standby state - a device in the online state is accessible to the software running in the partition or in the system, whereas a device in the standby state is no longer accessible.
- If referring to the PCICC facility as seen by ICSF, “online” means that the device is accessible but needs to have its Master Keys properly set. A PCICC is fully operational when in the “active” state, as described in 4.4.7 Management of the PCI Cryptographic Coprocessors.
- PCICA is seen by ICSF as “active” as soon as the PCICA state is set by CHPID to “online”, because there are no Master Keys in PCICA.

## 4.1 PCICC and PCICA card plugging

Keep in mind the following points:

- ▶ It is mandatory to follow the Installation Instructions provided with the MES, because part movements specific to the machine serial number are involved.
- ▶ The STI-attached PCICC/PCICA cards are installed in channel card slots, so the corresponding CHPID number must be free (that is, not used in the IOCDS definition); otherwise, a configuration conflict will occur.
- ▶ A PCICC card serial number is automatically associated to a free adjunct processor (AP) number, starting with number 0.

A PCICA card does not have serial number, but is also associated to a free adjunct processor (AP) number.

- ▶ When relocating a PCICC card to another system, the Master Keys are zeroized when the card is replugged. The customer has to reinstall the Master Keys.
- ▶ A card can be moved inside the system without changing its AP number.
- ▶ There is no explicit declaration of the PCICC/PCICA card to ICSF; ICSF has been notified of the physical presence of the PCICC by the operating system since OS/390 2.9. The operating system z/OS 1.2 and above will notify ICSF of the physical presence of a PCICA card (*only* on zSeries servers).

### 4.1.1 PCICC enablement

The following sections describe how to enable the PCICC.

**Note:** The PCICA does not have to be enabled because it does not contain any configuration data.

#### Function Control Vector (FCV) overview

- ▶ The Function Control Vector (FCV) enables the specific cryptographic functions with the authorized key lengths in the PCICC hardware.
- ▶ The FCV is shipped from manufacturing on a diskette, according to the requested feature code, when the first PCICC card is ordered or when the current cryptographic configuration needs to be changed.
- ▶ The PCICC FCV is specific to the system serial number and as such appears common to all PCI cards installed in the system.
- ▶ The installation of the FCV is a two-step process:
  - a. The FCV files are imported from the diskette onto the Support Element hard disk. This has to be done only when installing the first PCICC in the system or when changing the current cryptographic configuration. It can be done concurrently with system operations.
  - b. Then, using an option in the Support Element panel, the FCV is first uploaded to the system HSA, then downloaded into each PCICC card.
- ▶ If the FCV already resides in the HSA, it can be loaded or reloaded into the card concurrently. This is achieved by configuring the PCICC CHPID offline, then online at the Support Element (CHPID reset). To select a PCICC at the support element, do the following:
  - Right-click the CPC icon in the CPC Work Area. A drop-down menu appears with these options: CPs; CHPIDs; PCI Crypto.
  - Select **PCI Crypto**; this displays the PCICCs installed in the system.

From this point on you can refer to the *Support Element Operations Guide*, GC38-0608, for a description of how to configure a CHPID on or off.

## Import FCV

The Support Element must be logged on in SERVICE or SYSPROG, and the CPC logic must be powered on to get access to this panel.

- ▶ Select the CPC icon in the CPC Work Area.
- ▶ In the CPC configuration task list, select the task **PCI Cryptographic Coprocessor Configuration**.

The PCI Cryptographic Configuration main panel (see Figure 4-1) shows the following information about the installed PCI cards:

- ▶ Number (that is, the AP number)
- ▶ Status
- ▶ PCI serial number (PCICA cards does not have a serial number)
- ▶ Type (which is either PCICC or PCICA)
- ▶ UDX status (UDX is loadable only into PCICC)

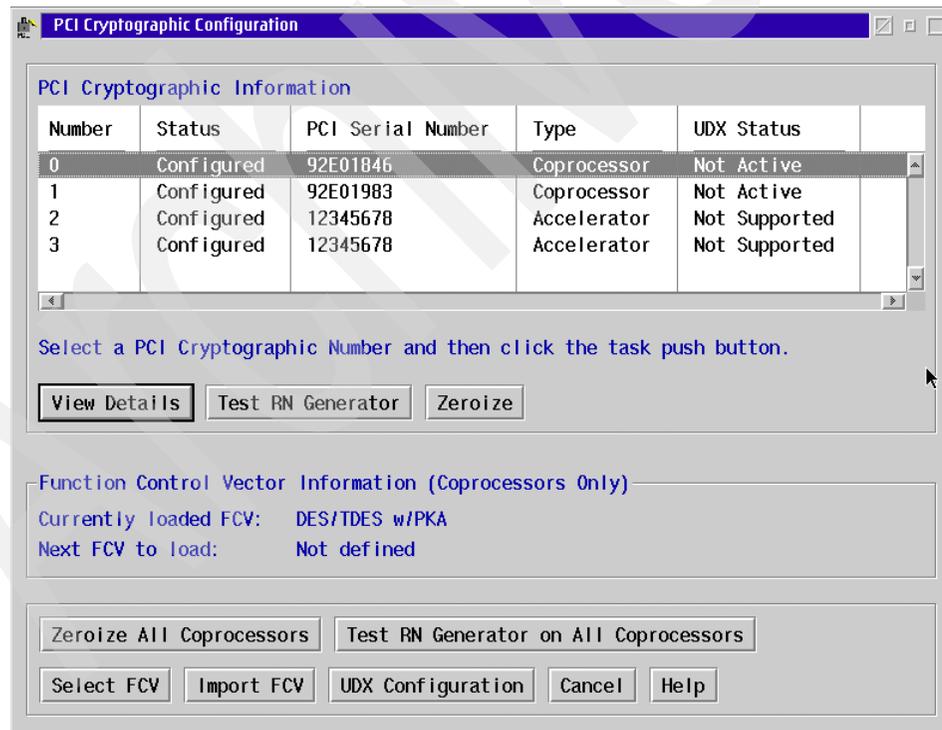


Figure 4-1 PCI Cryptographic Configuration main panel view

Selecting PCICC and pressing **View Details** in the PCICC main panel yields the panel shown in Figure 4-2.



Figure 4-2 PCI Cryptographic Coprocessor detail view

This view shows information about the AP number, CHPID, status, and the STI-attached Controller Card Serial Number. The PCICC card PCI serial number is shown in the ICSF panels. The details view also shows information about Segment 3 (see UDX enablement) and the Hash values for each segment.

Selecting PCICA and pressing **View Details** in the PCICC mail panel will show the panel in Figure 4-3 on page 106.



Figure 4-3 PCI Cryptographic Accelerator detail view

This view shows information about the AP number, CHPID, status, and the STI-attached Controller Card Serial Number. PCICA cards do not have an actual meaningful PCI card number.

In the PCICC main panel, you can at any time (if the CPC is powered on) press **Test RN Generator** to test only the selected PCICC, or **Test RN Generator on All Coprocessors**, which will run the same Random Number test on all PCICCs that are configured in the system. I

If the test works correctly, you will get the view shown in Figure 4-4 on page 107. (Note that this differs from the CCF, where ICSF must be started at least once before being able to run the pseudo random number generator test.)

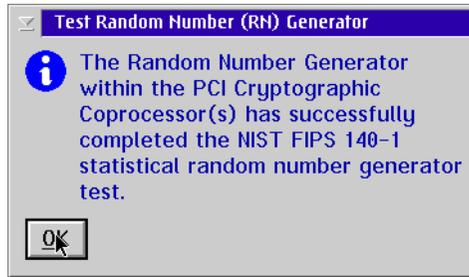


Figure 4-4 The result from the Random Number Generator test

Pressing the **Zeroize** button on the PCICC main panel (see Figure 4-1 on page 104) will zeroize all the cryptographic keys and configuration data and reset the authorities and profiles in the selected card. The zeroize command can also be sent to all the installed PCICCs.

**Note:** When the **Zeroize All Coprocessors** option is selected, it also erases the FCV loaded in HSA. Therefore, the FCV must be imported again, or the next attempt to initialize a PCICC card will result in the CHPID entering a check stop state.

When installing the first PCICC in the machine, or when changing the current FCV, you select the IMPORT FCV option to copy the FCV files from the diskette to the Support Element hard disk.

Then, after selecting the line of the target PCICC, using the option SELECT FCV will allow you to force the loading of the imported FCV into the PCICC hardware during the next activation (that is, PCICC CHPID configure off, then on, when changing the FCV, or when this is the first FCV imported into the system).

### Hardware status of the PCICC

To see the hardware status of the PCICC, you can double-click **PCI Cryptographic Coprocessor Configuration** on the CPC configuration task, when the CPC icon has been selected. Another possibility is to right-click the **CPC** icon in the CPC Work Area, which allows you to select CPs or CHPID or PCI Crypto, as shown in Figure 4-5 on page 108.

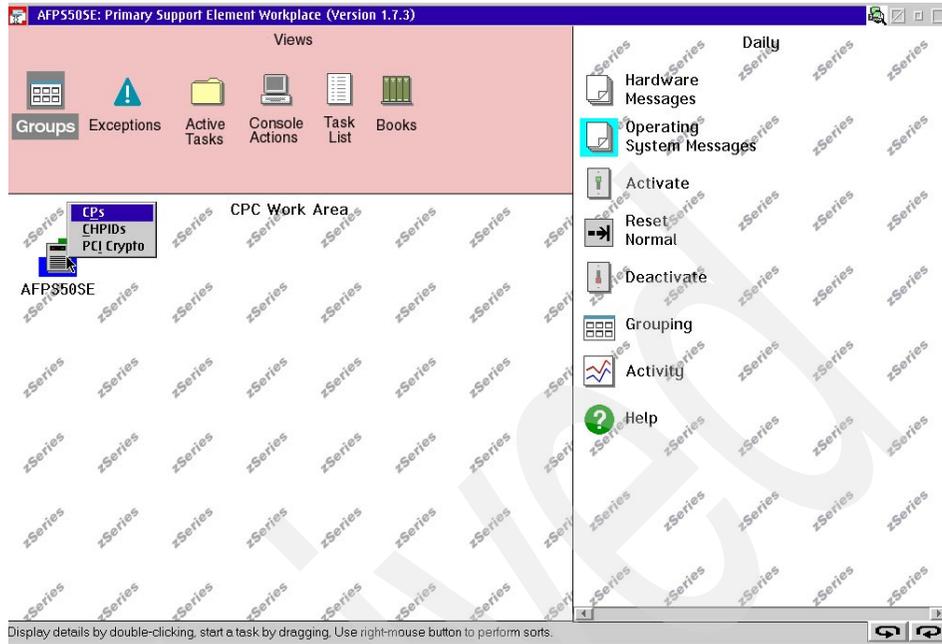


Figure 4-5 Right-clicked object on CPC Work Area view

When you select **PCI Crypto**, the panel shown in Figure 4-6 on page 109 displays all the PCI Crypto cards that are installed, and their status. A letter C or A indicates the type of the card (that is, C means PCICC and A means PCICA).

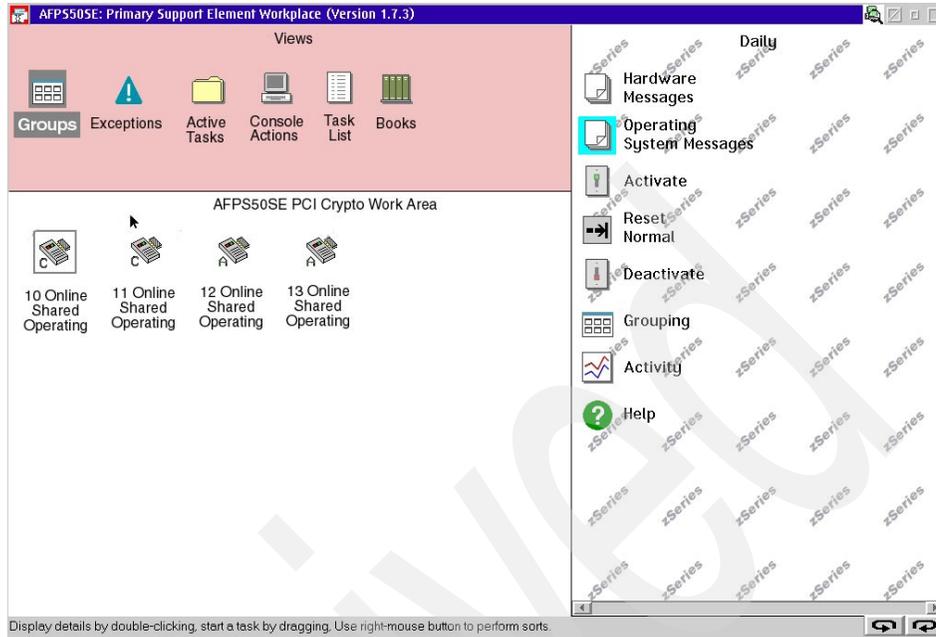


Figure 4-6 PCI Crypto selected from the CPC Work Area view

## 4.2 Installing User Defined Extensions (UDX)

Here we give an overview of how to install User Defined Extensions (UDX). The UDX code will be stored into Segment 3 area of the selected PCICC card.

**Important:** Storing the UDX code into a PCICC will zeroize all the cryptographic keys and configuration data and reset the authorities and profiles in the selected PCICC card. Therefore, the customer will have to reinstall the Master Keys and recreate the TKE cryptomodule customization for that PCICC.

To install the UDX code, you must have correctly packeted UDX code in the diskette. Log on to the Support Element as SERVICE or SYSPROG. (The CPC logic must be powered on to get access to this panel.)

1. Select the CPC icon in the CPC Work Area.
2. In the CPC configuration task list, select the task **PCI Cryptographic Coprocessor Configuration**.

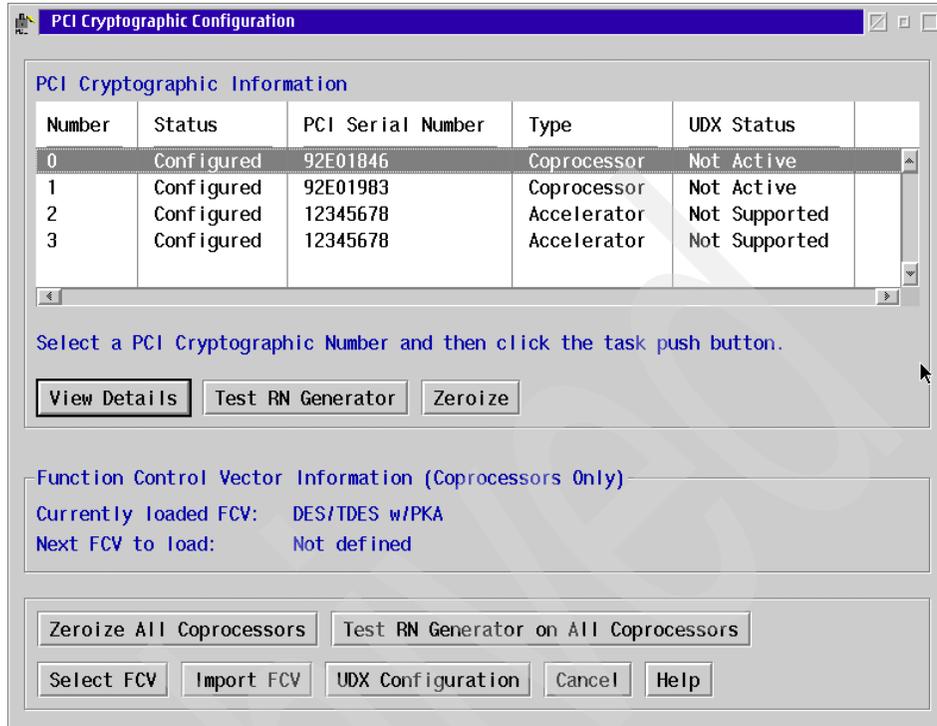


Figure 4-7 PCI Cryptographic Configuration main panel view

- From the PCI Cryptographic main panel view (Figure 4-7), select **UDX Configuration**.

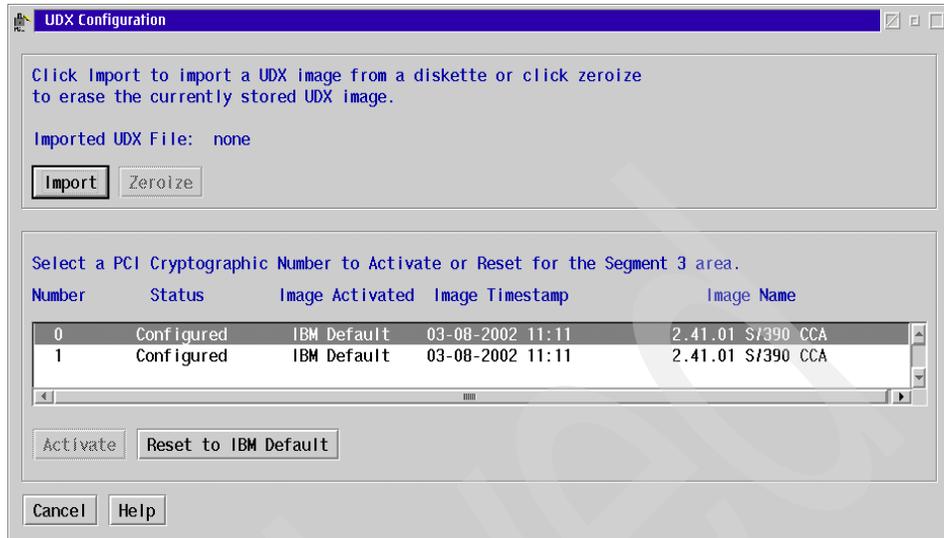


Figure 4-8 UDX Configuration page

- From the UDX Configuration panel view (Figure 4-8), select **Import** to copy the UDX code into Support Element hard disk (Figure 4-9).

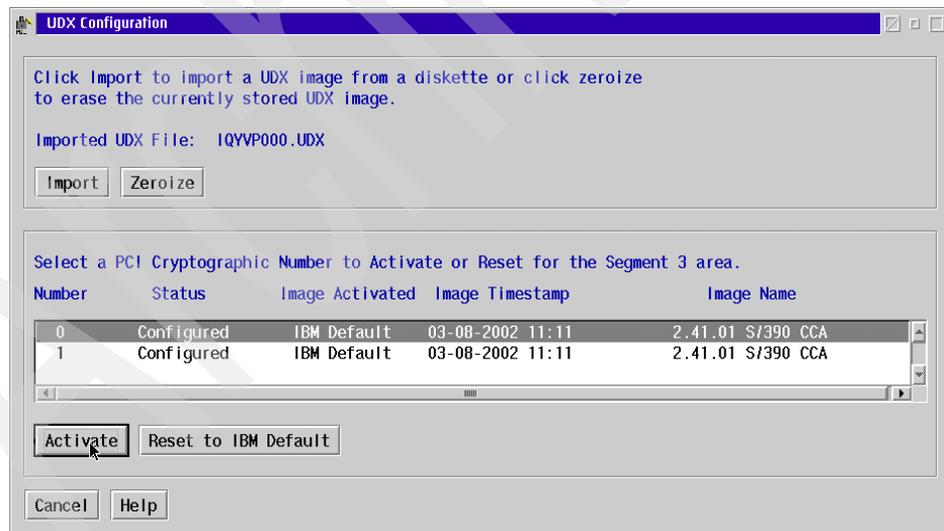


Figure 4-9 UDX Configuration page after UDX code import

5. After copying the UDX code, the name of the UDX file (in our case, IQYVP000.UDX) will appear on the UDX Configuration panel as seen in Figure 4-9.
6. Select the PCICC card where the code will be installed (only one PCICC card can be selected).
7. Press **Activate** to store the UDX code into the PCICC.

**Note:** After the UDX code is stored into PCICC, you will be asked if you want to zeroize the UDX code from the Support Element. Perform this task *after* the UDX code is stored into all desired PCICCs.

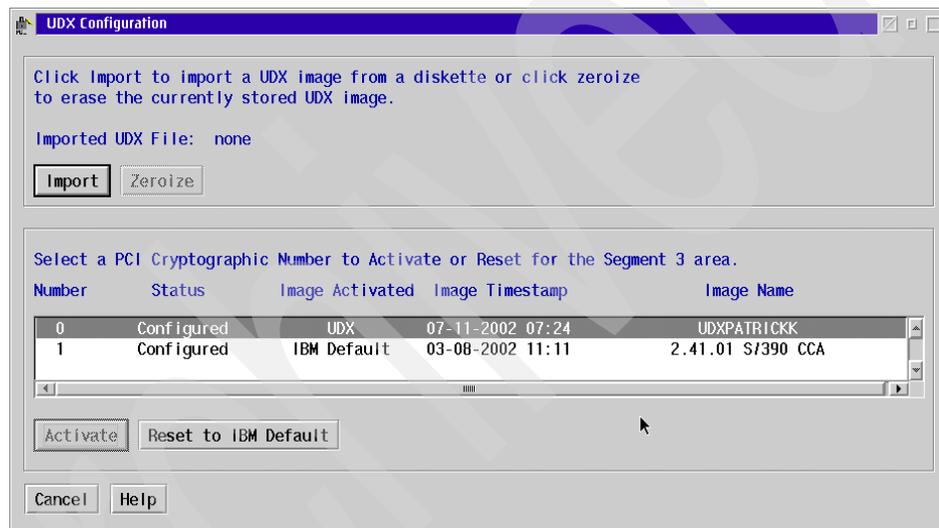


Figure 4-10 UDX Configuration panel after UDX code is stored into PCICC

- ▶ The UDX code Image Name (in our case, UDXPATRICKK) will appear in the UDX Configuration panel (Figure 4-10).

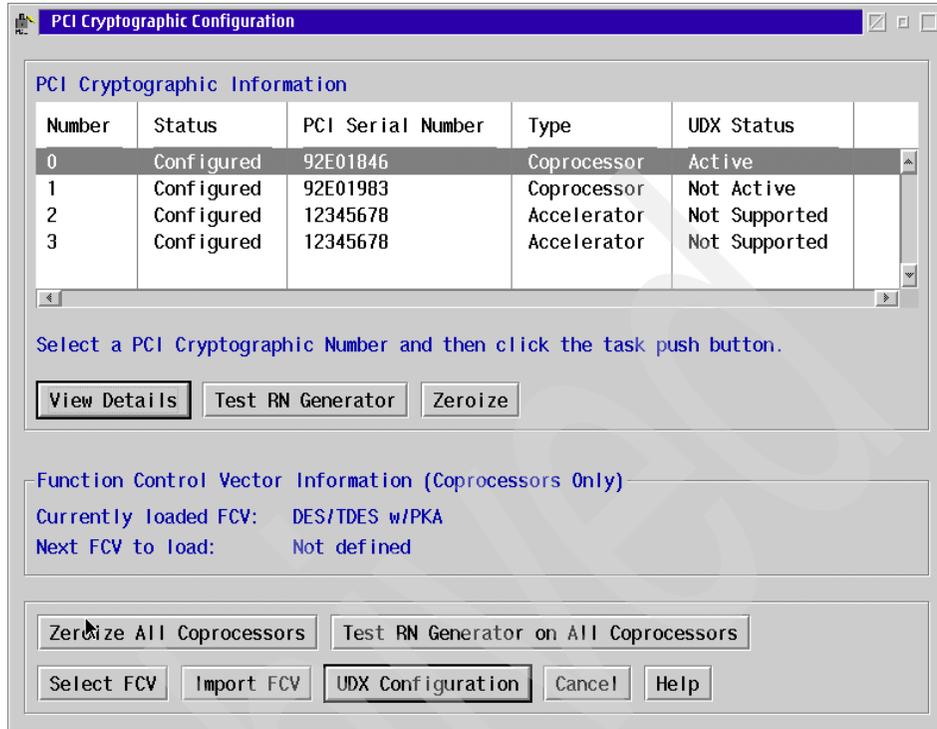


Figure 4-11 PCI Cryptographic Configuration main panel view

- ▶ From the PCI Cryptographic Configuration main panel (Figure 4-11) you can see that the UDX status has been changed to Active on the PCICC where the UDX code was stored.
- ▶ Selecting **View Details** will show the panel in Figure 4-12 on page 114.

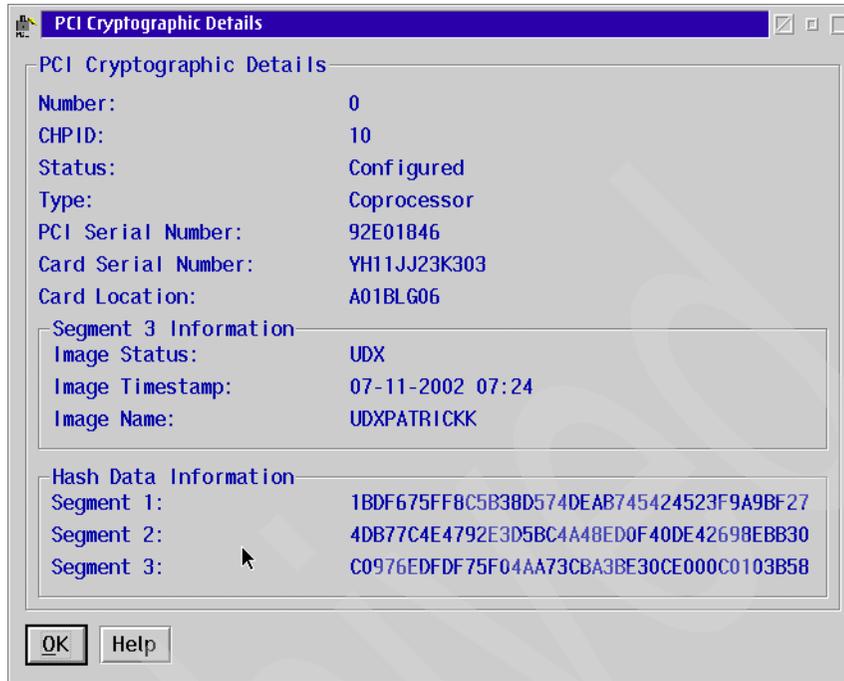


Figure 4-12 PCI Cryptographic Coprocessor detail view

- This view shows the changed information in Segment 3. The Hash Data Information shows that only Segment 3 has been changed (see Figure 4-2 on page 105).

## 4.3 LPAR setup

This section highlights the specific system setups when using the cryptographic coprocessors in a PR/SM environment.

The logical partition image profile is used by the system to set up the proper environment characteristics when activating the logical partition. The profile is made up of several pages and is kept recorded on the system Support Element hard disk. For further details on the contents of the image profile pages, refer to *PR/SM Planning Guide*, GA22-7236.

### 4.3.1 The image profile processor page

The processor page allows the selection of:

- ▶ The number of logical processors that will be made available to the logical partition
- ▶ Whether the partition will be dedicated or shared
- ▶ The number of logical crypto coprocessors that will be made available to the logical partition

Start invoking the customizing of the image profile for the logical partition by performing the following steps to get access to the processor page:

1. Select the **CPC** icon in the CPC Work Area view.
2. Double-click **Customize/Delete Activation Profile** in CPC Operational Customization. The panel shown in Figure 4-13 appears.

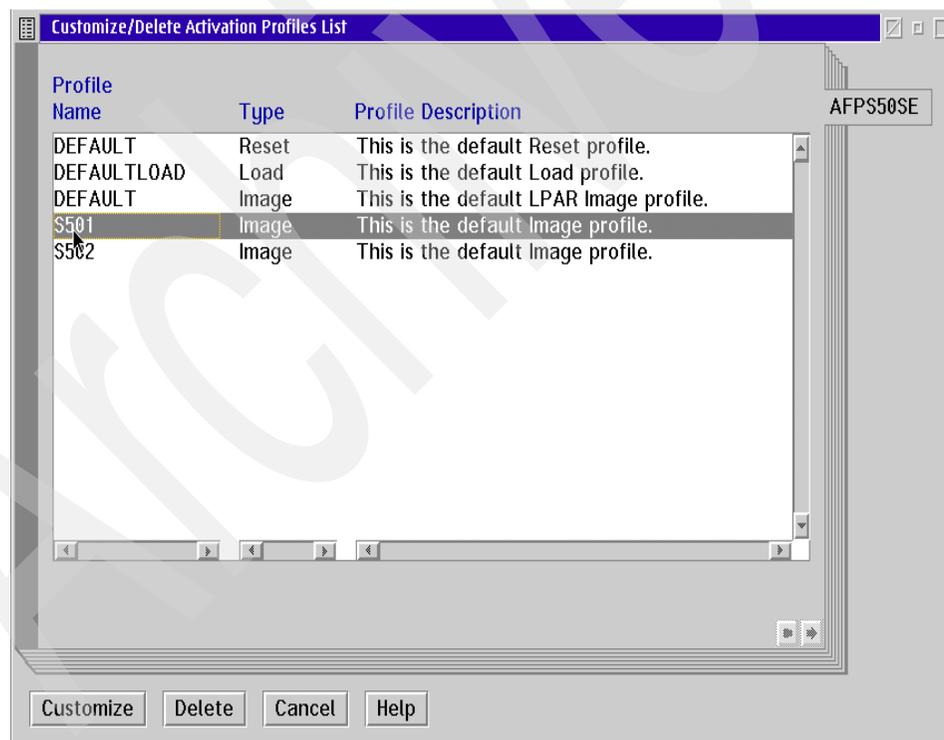


Figure 4-13 The Activation Profiles List

3. Select the image profile you want to work with and click **Customize**. In our case, we are working with the S501 image profile. The first page of the image profile is displayed, as shown in Figure 4-14.

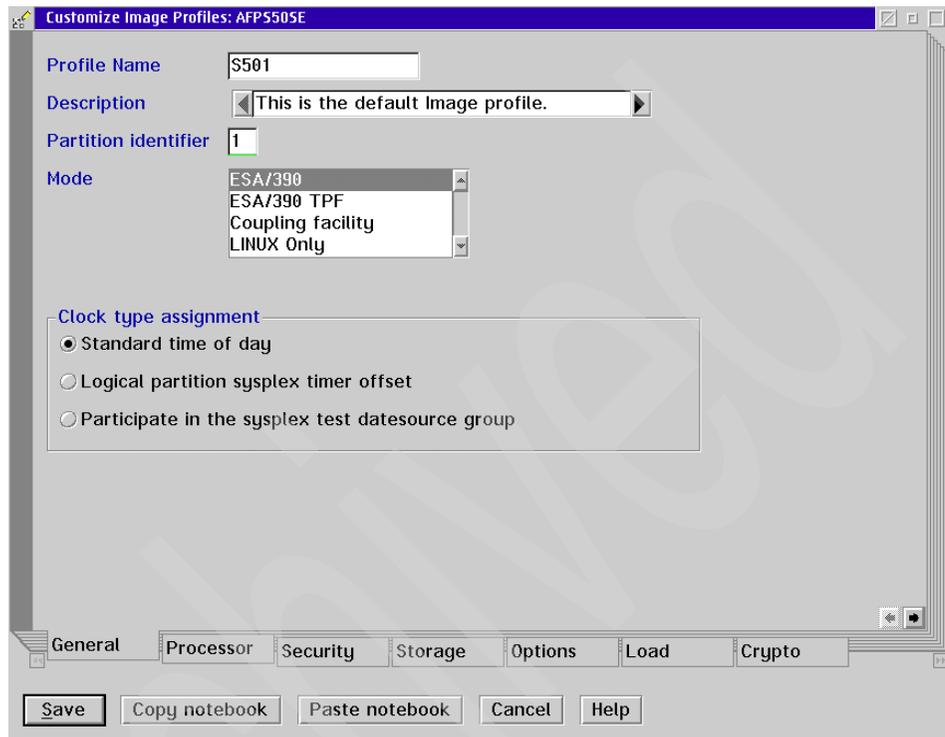


Figure 4-14 Customize Image Profile page of a selected image

4. Select the **Processor** tab to see the processor page; see Figure 4-15 on page 117.

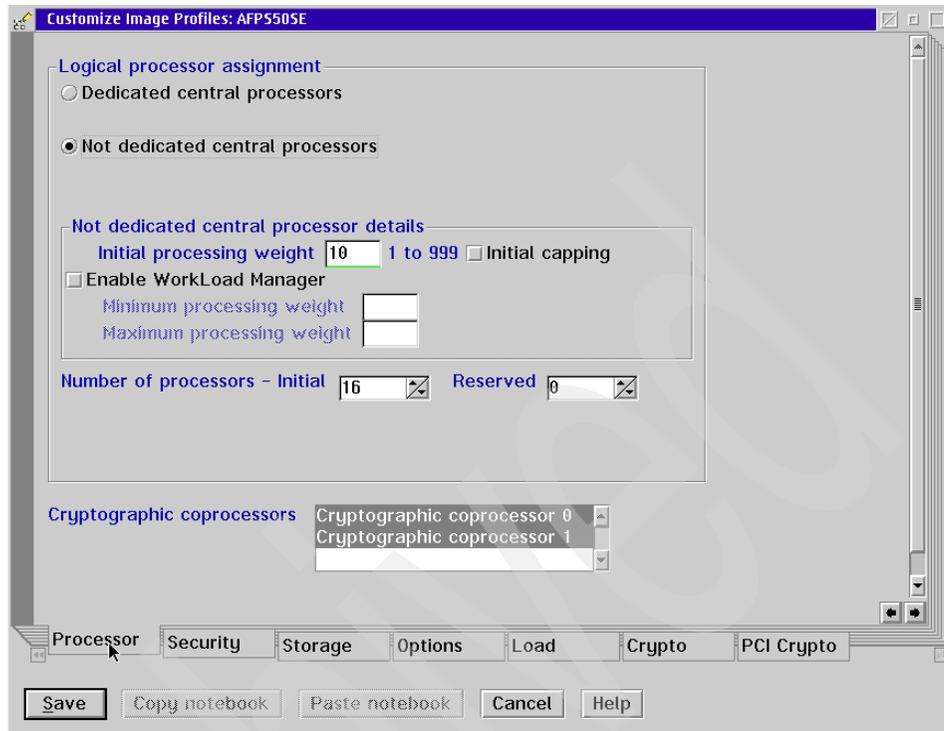


Figure 4-15 Customizing Image Profile: the processor page

The Image Profile processor page shown in Figure 4-15 contains the Cryptographic coprocessors window, indicating how many CCFs are physically available in the system. Selecting one or two of these chips allocates one or two logical crypto coprocessors for the logical partition.

These logical crypto coprocessors will actually be backed up by physical resources only when the logical partition is activated. Note that there are initially no Crypto and PCI Crypto tabs offered for selection at the bottom of the pages. As soon as a logical processor has been allocated to the partition, the Crypto and PCI Crypto tabs appear, as shown in Figure 4-15 on page 117.

### Dedicated central processors

Electing to run a logical partition with dedicated central processors *and* cryptographic coprocessors has a very dramatic effect on the rest of the system when this logical partition is activated, because there is a physical affinity between the crypto chips and two physical processors in the system.

The PR/SM microcode will allocate one or two of these physical processors, depending on the number of logical cryptographic coprocessors selected, to the

dedicated logical partition, thus making these physical processors unavailable to other logical partitions. This can also result not being able to activate the dedicated partition because of other shared partitions already sharing the physical crypto chips.

**Important:** The concept of coprocessor dedication does not apply to PCICC. A PCICC is by nature accessible from several partitions as directed by the online and candidate lists, regardless of the shared or dedicated status given to the CCF.

### 4.3.2 The Crypto page

The Crypto page allows the selection of parameters for the cryptographic setup of the logical partition. These setup parameters are taken into account at logical partition activation.

Note that these selections, except for the contents of the control and usage domain lists, can be dynamically changed without needing to deactivate and then reactivate the logical partition, by using the “Change LPAR Cryptographic Controls” in the CPC Operational Customization task list.

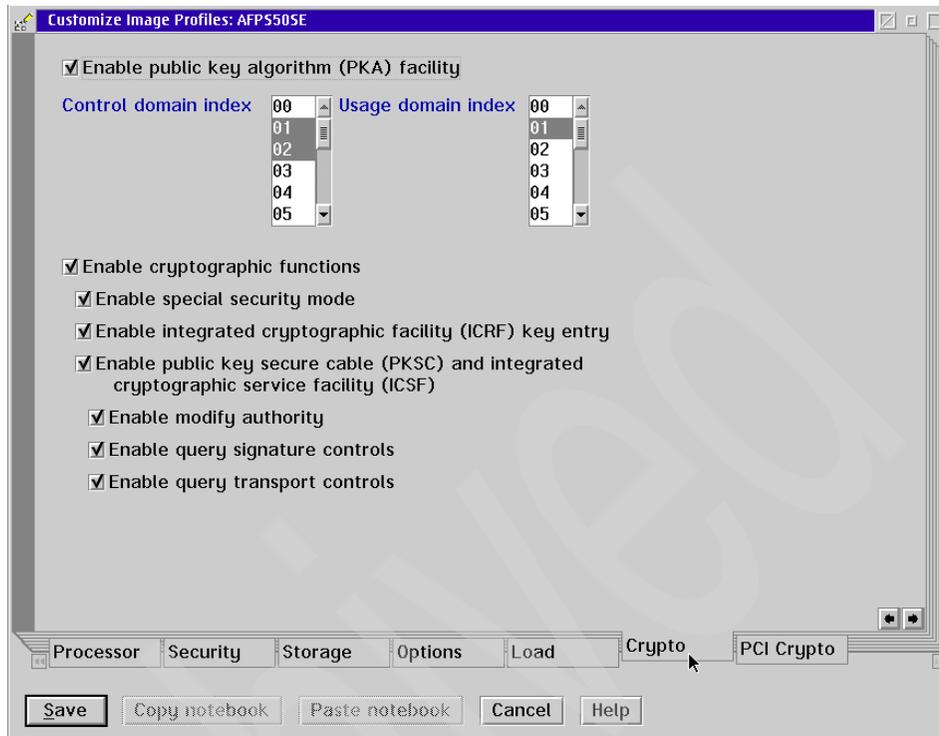


Figure 4-16 Crypto page of the image profile

The selections shown in Figure 4-16 were made on the Crypto page of the image profile of our S501 partition.

- ▶ **Enable public key algorithm (PKA) facility** - The original intent of this selection was to select PKA-only functions, as opposed to the DES-based functions. Although this option seems to imply that you may also have the choice of not enabling your logical crypto coprocessors for PKA types of functions, this selection is *mandatory* in order to have ICSF start correctly.
- ▶ **Usage domain index** - This must point to the domain number(s) allowed to this logical partition, which should also match the domain number(s) entered in the Options dataset when starting this partition's instance of ICSF.

This is the domain number that you are giving to this partition. We recommend that you choose a number equal to the partition ID for a clear relationship between logical partition and domain.

**Note:** Beginning with z/OS1.2, the Usage Domain information is optional in the Options dataset if only one Usage Domain is defined for the logical partition. If more than one domain is specified as the Usage Domain on this panel, or if running ICSF in native mode, then domain information is required in the Options dataset.

When there are multiple domains specified in the Usage Domain and no information is in the Options dataset, you will get the following message in the system log after ICSF is started:

```
$HASP373 CSF      STARTED
CSFM409E MULTIPLE DOMAINS AVAILABLE. SELECT ONE IN OPTIONS DATA SET.
$HASP395 CSF      ENDED
```

- ▶ **Control domain index** - This pertains to the logical crypto coprocessors defined in other logical partitions that can be administered from this logical partition being set up as the TCP/IP host for the TKE.

If you are setting up the host TCP/IP in this logical partition for communicating with the TKE, this is the partition that will be used as a path to other domains' Master Keys. Indicate all the domains you want to access (including this partition's own domain) from this partition as control domains. In the example shown in Figure 4-16 on page 119, you are able to access Master Keys for domains 1 and 2 from the TKE.

- ▶ **Enable cryptographic functions** - This actually pertains to the DES-based ICRF functions (as they were known in the ES/9000-9021), but also includes the DES Master Key management functions.

It must be selected for normal operation of the logical cryptographic coprocessors.

- ▶ **Enable special security mode** - This is the mode of operation required for KGUP to permit generation or entry of clear keys, and to enable the secure key import or clear pin generate callable services. ICSF will work with the clear keys (as opposed to keys encrypted under a Master Key or a Transport Key) only if Special Security Mode (SSM) is selected on this page of the image profile and also specified as SSM YES in the Options data set. Only deselecting this option in the image profile is enough to prevent ICSF from accepting clear keys. Note that the status of the SSM can also be modified for the domain at the TKE. This is accomplished by setting the SSM bit in the ECM.

The setting of this selection is actually a matter of security policies and procedures at the installation. Refer to these customer procedures to decide what to do.

- ▶ **Enable integrated cryptographic facility (ICRF) key entry** - This option must be selected to permit the loading of Master Keys in the usage domain from the TKE. No selection implies that the Master Keys can only be entered from the ICSF ISPF panels. Select whether you want to use the TKE to enter Master Keys into the usage domain of this partition.
- ▶ **Enable public key secure cable (PKSC) and integrated cryptographic service facility (ICSF)** - Do not be misled, there is no choice here: this *must* be selected. Furthermore, it gives access to the last three options.
- ▶ **Enable modify authority** - There can be only one active logical partition at one time with this option selected. The definitions of authority and authority signatures will be transmitted into the physical cryptographic coprocessor from the TKE through this logical partition, if this is selected. Refer to the security policies and procedures for the installation before making this selection.
- ▶ **Enable query signature controls** - This enables the TKE to get, through this logical partition, the status of the multi-signature command in progress; in other words, what are the multi-signature commands defined by the installation? What is the one in progress? Has the crypto coprocessor received all the necessary signatures? Refer to the security policies and procedures for the installation.
- ▶ **Enable query transport controls** - This allows the logical partition to participate in a key transfer operation with the TKE Workstation. Refer to the security policies and procedures for the installation.

### 4.3.3 The PCI Crypto page

The PCI Crypto page (see Figure 4-17 on page 122) allows for the selection of the AP numbers to be put in the online and candidate lists.

- ▶ **PCI Cryptographic Coprocessor Candidate List** - This pertains to the AP numbers that are accessible for this logical partition, but must be manually brought online to the partition.
- ▶ **PCI Cryptographic Coprocessor Online List** - This pertains to the APs that are automatically brought online during logical partition activation. If not all the selected APs are online, an activation error condition is reported. The APs selected in the Online List must also be part of the Candidate List.

**Note:** The PCI Cryptographic Coprocessor Online and Candidate List selection cannot be changed dynamically; that is, any change to these lists requires deactivating and then activating the logical partition to be taken into account. In other words, addition of a PCICC in the system must be planned ahead of LPAR activation in order to be truly concurrent.

Before installing the PCICC and PCICA cards, we initially made the following selections on the PCI Crypto page of the image profile (see Figure 4-17). This gave us the opportunity to install the PCICC and PCICA cards concurrently.

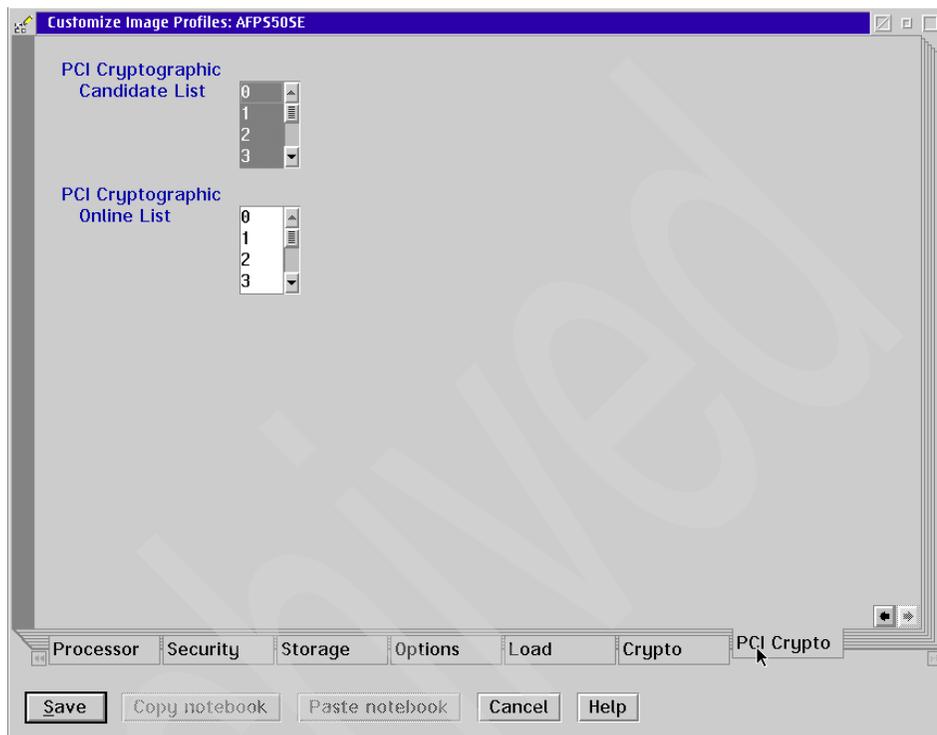


Figure 4-17 PCI Crypto page of the image profile

If there is a need to increase the amount of PCICC/PCICA cards later, we recommend that you select all in the PCI Cryptographic Candidate List. This can be seen as a preparation for the concurrent installation of the additional PCI cards.

**Important:** After LPAR deactivation and activation, all PCI cards that are on the PCI Cryptographic Candidate List but not on the PCI Cryptographic Online List are in a “configured off state (Standby)”, and they can be configured on *only* by using the Support Element or Hardware Management Console from the CHPID Operations panel, using the “Configure On/Off” selection.

## PCICC and cryptographic domains

The AP identification number, from 0 to 15, is used when defining the PCICC or PCICA cards on the PCI Crypto page. Only one PCICC card installed into the

system has a separate Cryptographic processor that has its own Domains, from 0 to 15.

It is not possible to assign PCICC Domain indices to a logical partition. The CCF domain indices in the Crypto page (control and usage domains) are propagated to the PCICC. ICSF accesses the CCF and PCICC Domain whose index value is given in the Options dataset (if given), provided that this domain index value is in the domain usage list for the partition.

#### 4.3.4 Changing LPAR Cryptographic controls dynamically

Certain LPAR Cryptographic Control definitions can be changed dynamically (that is, without deactivation/activation of the LPAR). This possibility can be used, for example, to toggle “Enable special security mode” to allow/disallow clear key process.

To invoke customization of the LPAR Cryptographic Controls, follow these steps to get access to the page:

1. Select the **CPC** icon in the CPC Work Area view.
2. Double-click **Change LPAR Cryptographic Controls** in CPC Operational Customization. The panel shown in Figure 4-18 on page 124 appears.

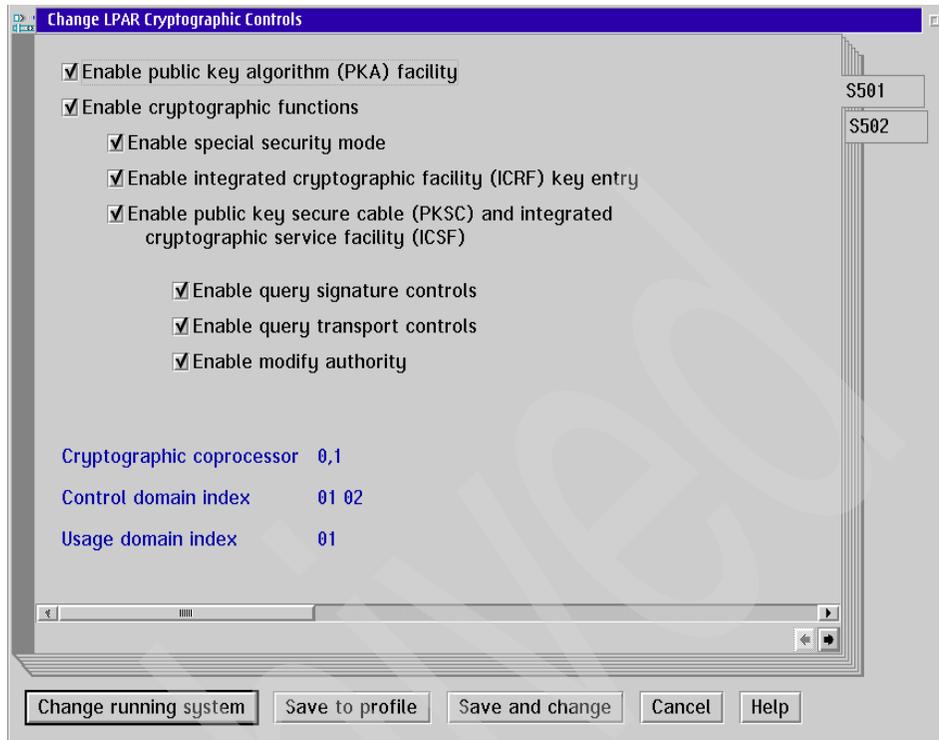


Figure 4-18 Change LPAR Cryptographic Controls page

Make the change on the panel and select the desired action from the bottom of the panel (for example, to change LPAR Cryptographic Controls permanently, select the **Save and change** option, which makes the change to the running system and also saves the change into the LPAR profile).

## 4.4 Integrated Cryptographic Services Facility (ICSF) setup

In the following sections we describe how to perform ICSF setup, as well as the major changes from the previous releases.

## 4.4.1 Major changes from previous releases

### **PCI Cryptographic Accelerator (PCICA) support**

PCICA support has been added on z/OS 1.2. If a PCI Cryptographic Accelerator is available, clear RSA key processing in the CSFDPKD service will be routed to the PCI Cryptographic Accelerator.

### **PKDS re-encipherment**

Starting from z/OS 1.2, the PKDS has been able to re-encipher. This feature requires at least one PCICC to be installed and enabled in the system. To enable PKDS re-encipher to work properly, keep the same Master Key value on the CCF PKA SMK, KMMK, and the PCICC ASYM-MK.

### **User Defined Extensions (UDX) support**

Beginning with OS/390 2.10 ICSF, ICSF support is provided for UDX capabilities. UDX routines are developed by special contract with IBM and are only distributed to authorized customers. Since z/OS 1.2, support for writing the customer's own UDX has been added.

### **RMF™ support**

z/OS 1.2 supported RMF to provide limited reports about the performance of the Cryptographic hardware. On z/OS 1.3, RMF has been enhanced to provide performance reports on selected ICSF services and functions that use CCF instructions.

### **Options Dataset changes**

Starting with z/OS 1.2, the DOMAIN parameter is an optional parameter. It is, however, required if more than one domain is specified as the usage domain on the PR/SM panels, or if running in native mode.

Also since z/OS 1.2, MAXLEN parameter checking has been eliminated for the following services:

- ▶ Encipher/decipher
- ▶ MAC generate/verify
- ▶ Ciphertext translate
- ▶ MDC generate

The MAXLEN parameter may still be specified in the options dataset, but only the maximum value limit will be enforced (2147483647).

Again since z/OS 1.2, PKDSCACHE parameter could be used to define the size of the PKDS Cache in records. The PKDS cache improves performance, as it

facilitates access to frequently used records. If PKDSCACHE is not specified, the default value is 64.

### ICSF TSO panels

Starting from z/OS 1.3, the ICSF TSO panels have been updated to enhance usability:

- ▶ Coprocessor management functions have been combined onto one panel.
- ▶ Master Key management/CKDS functions have been combined onto one panel.
- ▶ TKE TSO utilities have been combined onto one panel.
- ▶ The Primary panel has been simplified.
- ▶ A new utility has been added to generate Master Key values from a pass phrase.

### AES support

Beginning with z/OS 1.3, new callable services has been added to support AES encryption/decryption. AES encryption is only allowed if Triple DES is enabled. Only clear key support is provided.

### EMV support

Beginning with z/OS 1.2, the following functions are supported for EMV:

- ▶ New Diversified Key Generate (CSNBKDG)
- ▶ Additional support for generation of RSA keys under KEK (CSNDPKG)
- ▶ Additional support in Digital Signature Gen (CSNDDSG) and verify (CSNDDSV)
- ▶ Secure messaging (ISO 7816-4) for keys and PINs (CSNBSKY, CSNBSPN).

## 4.4.2 Started task and the first time start

To protect the ICSF started task user ID from being revoked through malicious or inadvertent incorrect password attempts, we recommend that you use a protected user ID, that is, one with the NOPASSWORD attribute. For more information, see *Security Server (RACF) Security Administrator's Guide*, SC28-1915, OS/390 2.8 or later.

Access to the ICSF administrative tasks or services can be restricted using RACF Facility class profiles.

When you start ICSF for the first time, you will see the messages shown in Figure 4-19 on page 127. You receive the message CSFM511E for each Cryptographic Coprocessor Feature you have online.

```

S CSF
£HASP100 CSF      ON STCINRDR
IEF695I START CSF      WITH JOBNAME CSF      IS ASSIGNED TO USER SYSTASK
, GROUP SYS1
£HASP373 CSF      STARTED
CSFM100E CRYPTOGRAPHIC KEY DATA SET, PATKAP.CKDS3 IS NOT INITIALIZED.
CSFM511E CRYPTOGRAPHY - MASTER KEY ON COPROCESSOR C0, CPU 0 IS NOT
VALID.
CSFM511E CRYPTOGRAPHY - MASTER KEY ON COPROCESSOR C1, CPU 1 IS NOT
VALID.
CSFM106A CRYPTOGRAPHY - PKA MASTER KEYS ARE NOT VALID.
CSFM009I NO ACCESS CONTROL AVAILABLE FOR ICSF SERVICES OR KEYS
CSFM001I ICSF INITIALIZATION COMPLETE

```

Figure 4-19 The first-time ICSF startup messages

### 4.4.3 Master Keys

ICSF uses three Master Keys to protect the keys that are used with CCF, and two Master Keys to protect the keys that are used with the PCICC.

#### Master Keys in the CCF

##### *The DES Master Key*

This key is a double-length (128-bit) key that is used to protect DES and CDMF keys by encrypting these keys with the triple DES algorithm.

##### *The PKA Key Management Master Key*

This key (KMMK) is a triple-length (192-bit) key. The KMMK protects PKA private keys that are used in both the digital signature services and in the CDMF and DES data key distribution functions.

##### *The PKA Signature Master Key*

This key (SMK) is a triple-length (192-bit) key. The SMK protects PKA private keys that are used only in digital signature services.

#### Master Keys in the PCICC

##### *Symmetric-keys Master Key*

This key (SYM-MK) is a double-length (128-bit) key that is used to protect DES keys used on the PCI Cryptographic Coprocessor. The SYM-MK is actually a triple-length (192-bit) Master Key inside the PCICC that ICSF enforces to be equivalent to a double-length (128-bit) Master Key. This key must have the same value as the DES Master Key in the CCF.

### **Asymmetric-keys Master Key**

This key (ASYM-MK) is a triple-length (192-bit) key. The ASYM-MK protects RSA private keys that are used on the PCI Cryptographic Coprocessor. This key must have the same value as the SMK in the CCF.

#### **4.4.4 Initial Master Key entry with the pass phrase initialization utility**

The pass phrase initialization utility can be used to initialize the CKDS and the coprocessor. You can also use this utility to install both the SYM-MK and the ASYM-MK on all PCI Cryptographic Coprocessors on S/390 G5 Enterprise Servers, or later.

To use this utility, the special secure mode must be enabled, and all the Master Key registers in the targeted coprocessors must be empty. Also, the designated CKDS must not be already initialized, that is, it is just an empty VSAM data set. Therefore, you cannot use this utility to change the Master Keys.

When you access the ICSF panels, the Primary menu panel appears, as shown in Figure 4-20.

```
----- Integrated Cryptographic Service Facility-----
Enter the number of the desired option.

  1 COPROCESSOR MGMT - Management of Cryptographic Coprocessors
  2 MASTER KEY      - Master key set or change, CKDS/PKDS Processing
  3 OPSTAT          - Installation options
  4 ADMINCNTL      - Administrative Control Functions
  5 UTILITY         - ICSF Utilities
  6 PPINIT         - Pass Phrase Master Key/CKDS Initialization
  7 TKE            - TKE Master and Operational Key processing
  8 KGUP           - Key Generator Utility processes
  9 UDX MGMT       - Management of User Defined Extensions

Licensed Materials - Property of IBM
This product contains "Restricted Materials of IBM"
5694-A01 (C) Copyright IBM Corp. 2002. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Press ENTER to go to the selected option.
Press END  to exit to the previous menu.
OPTION ==> 6
```

*Figure 4-20 Selecting Pass Phrase Option on the ICSF Primary menu panel*

To begin the pass phrase initialization utility, select option **6, PPINIT**, and press Enter. Then type the pass phrase and the data set name in the spaces provided.

Answer **Y** to the question Signature MK = Key Management MK? As a result, the signature Master Key and the Key Management Master Key in the CCFs will have the same value as the ASYM-MK on the PCI Cryptographic Coprocessors.

Press Enter to run the utility. Its progress is indicated by messages, as shown in Figure 4-21.

```
----- ICSF - Pass Phrase MK/CKDS Initial  INITIALIZATION COMPLETE

Enter your pass phrase and the name of the CKDS:

Pass Phrase (16 to 64 characters)
===> zseries crypto guide update

CKDS
===> 'PATKAP.CKDS3'

Initialize the CKDS? (Y/N) ===> Y
Signature MK = Key Management MK? (Y/N) ===> Y
Initialize new PCICCs only ? (Y/N) ===> N

The master key registers have been loaded.
Processing of the CKDS is complete.
Pass phrase initialization has been completed.

Press ENTER to process.
COMMAND ===>
```

Figure 4-21 Pass phrase initialization complete messages

After pass phrase initialization, messages in the system log show that both cryptos are online and ICSF is ready for work (see Figure 4-22).

```
IEE504I CRYPTO(0),ONLINE
IEE504I CRYPTO(1),ONLINE
CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.
```

Figure 4-22 System log messages after pass phrase

## Hardware status display after pass phrase initialization

To display the hardware status using the ICSF/ISPF panels:

1. Select option **1, COPROCESSOR MGMT**, on the ICSF Primary menu panel; you will get the panel shown in Figure 4-23.

```
----- ICSF Coprocessor Management ----- Row 1 to 2 of 2

Select the coprocessors to be processed and press ENTER.
Action characters are: A, D, E, R, and S. See the help panel for details.

      COPROCESSOR      MODULE ID/SERIAL NUMBER      STATUS
      -----      -----
.   C0                04100000000039C4 04100000000039C4  ACTIVE
.   C1                0410000000003992 0410000000003992  ACTIVE
***** Bottom of data *****
```

Figure 4-23 ICSF Coprocessor Management panel view

2. From this new panel, select the processor(s) that has the content that you want to see by entering S or / in front of the processor line, then press Enter.
3. The scrollable Coprocessor Hardware Status panel appears. On this panel, you can view the status of the Cryptographic Coprocessor Feature. You can check whether a unit is active and whether its registers are in the correct state.

The key register fields show the state of the registers and their verification pattern and/or hash pattern values.

The PKA Key Management Master Key register hash pattern and the PKA Signature Master Key register hash pattern would also be equal in our case, which was the requirement when we started the pass phrase initialization utility.

### 4.4.5 Installation of the PCICC and PCICA cards

After concurrent installation of the PCI Cryptographic Coprocessor (PCICC) and PCI Cryptographic Accelerator (PCICA) cards, both PCICA cards become active because they are working with clear key cryptography and they are not using ICSF Master Keys.

PCICC cards must have the same DES Master Key as is installed on the CCF. The ASYM-MK must have the same value as the SMK on the CCF in order to

share the same PKDS. Also, the re-encipherment of the PKDS will work correctly only if these keys have the same value.

**Important:** We strongly suggest that CCF's SMK, KMMK, and PCICC's ASYM-MK all have the same value.

To enter the same Master Key values to the newly installed PCICC cards, the Pass Phrase initialization panel has a new option to initialize only new PCICCs. From the ICSF main panel, we select option **6, PPINIT**.

On the Pass Phrase initialization panel, we enter the same pass phrase as before. The name of the CKDS must be filled in. This time, we only respond Y to the question of initializing new PCICCs only. After pressing Enter, PCICC initialization will be complete, as shown in Figure 4-24.

```
----- ICSF - Pass Phrase MK/CKDS Initial  INITIALIZATION COMPLETE

Enter your pass phrase and the name of the CKDS:

Pass Phrase (16 to 64 characters)
===> zseries crypto guide update

CKDS
===> 'PATKAP.CKDS3'

Initialize the CKDS? (Y/N) ===> N
Signature MK = Key Management MK? (Y/N) ===> Y
Initialize new PCICCs only ? (Y/N) ===> Y
```

Figure 4-24 Pass Phrase initialization of the PCICCs

System log messages show that the entered Master Keys match and both PCICC cards are active; see Figure 4-25.

```
CSFM116I BOTH MASTER KEYS CORRECT ON PCI CRYPTOGRAPHIC COPROCESSOR P00,
SERIAL NUMBER 92E01846.
CSFM116I BOTH MASTER KEYS CORRECT ON PCI CRYPTOGRAPHIC COPROCESSOR P01,
SERIAL NUMBER 92E01983.
```

Figure 4-25 Both PCICC cards become active after Pass Phrase initialization

To view the Hardware Status after PCICC and PCICA installation, select option **1, COPROCESSOR MGMT**, on the ICSF Primary menu panel; you will get the panel shown in Figure 4-26.

```

----- ICSF Coprocessor Management ----- Row 1 to 6 of 6

Select the coprocessors to be processed and press ENTER.
Action characters are: A, D, E, R, and S. See the help panel for details.

      COPROCESSOR      MODULE ID/SERIAL NUMBER      STATUS
      -----      -
.   A02
.   A03
.   C0          04100000000039C4 04100000000039C4  ACTIVE
.   C1          0410000000003992 0410000000003992  ACTIVE
.   P00          92E01846
.   P01          92E01983
***** Bottom of data *****

```

Figure 4-26 PCICA and PCICC installed

To display the status of the PCICCs or CCF, select the cryptographic processor by entering *S* or */* in front of the processor line. The coprocessor type is shown by the letter in coprocessor column, where *A* means PCICA, *C* means CCF, and *P* means PCICC. The number, together with *A* and *P*, points to the AP numbers shown in Figure 4-17 on page 122.

#### 4.4.6 Changing the PKA Master Keys via ICSF panels

Beginning with z/OS 1.2, the PKDS has been able to re-encipher. This feature requires at least one PCICC to be installed and enabled in the system. Use PKDS re-encipher to convert to a system where all the PKA MKs are the same.

To change the PKA Master Keys, go to the Administrative Control Functions by selecting option **4, ADMINCNTL** on the Primary menu panel (see Figure 4-20 on page 128) to first disable the PKA Callable Services; this must be done before changing the PKA Master Keys (see Figure 4-27 on page 133).

```

----- ICSF - Administrative Control Functions -- Row 1 to 4 of 4

Active CKDS: PATKAP.CKDS4
Active PKDS: PATKAP.PKDS4

To change the status of a control, enter the appropriate character
(E - ENABLE, D - DISABLE) and press ENTER.

FUNCTION                                STATUS
-----                                -----
. Dynamic CKDS Access                   ENABLED
d PKA Callable Services                 ENABLED
. PKDS Read Access                      ENABLED
. PKDS Write, Create, and Delete Access  ENABLED
***** Bottom of data *****

```

Figure 4-27 Disabling PKA Callable Services

On the Administrative Control Functions panel, enter d next to PKA Callable Services and press Enter; PKA Services will be disabled, as shown in Figure 4-28.

```

----- ICSF - Administrative Control Fun   PKA SERVICES DISABLED

Active CKDS: PATKAP.CKDS4
Active PKDS: PATKAP.PKDS4

To change the status of a control, enter the appropriate character
(E - ENABLE, D - DISABLE) and press ENTER.

FUNCTION                                STATUS
-----                                -----
. Dynamic CKDS Access                   ENABLED
. PKA Callable Services                 DISABLED
. PKDS Read Access                      DISABLED
. PKDS Write, Create, and Delete Access  DISABLED
***** Bottom of data *****

```

Figure 4-28 PKA Services disabled

Before entering the new PKA Master Keys, we have to reset the PKA Master Key register. This is done by selecting **COPROCESSOR MGMT** on the ICSF Primary menu panel, and then selecting which Coprocessors should have their PKA Master Keys reset (see Figure 4-29).

```

----- ICSF Coprocessor Management ----- Row 1 to 6 of 6

Select the coprocessors to be processed and press ENTER.
Action characters are: A, D, E, R, and S. See the help panel for details.

  COPROCESSOR      MODULE ID/SERIAL NUMBER      STATUS
  -----
.  A02
.  A03
e  C0              04100000000039C4 04100000000039C4  ACTIVE
e  C1              0410000000003992 0410000000003992  ACTIVE
.  P00              92E01846              DEACTIVATED
e  P01              92E01983              ACTIVE
***** Bottom of data *****

```

Figure 4-29 Selecting processors for resetting the PKA Master Key register

To get all the PKA Master Keys reset, we selected **ALL-PKA** for the Key Type (see Figure 4-30 on page 135).

```
----- ICSF - Clear Master Key Entry -----
COMMAND ==>

          CCF DES/PCICC SYM-MK new master key register   : EMPTY
          CCF Signature/PCICC ASYM-MK master key register : NOT THE SAME
          CCF Key Management master key register          : NOT THE SAME

Specify information below

Key Type ==> all-pka          (DES, SMK, KMMK, ALL-PKA)
Part     ==> reset           (RESET, FIRST, MIDDLE, FINAL)
Checksum ==> 00

Key Value ==> 0000000000000000
          ==> 0000000000000000
          ==> 0000000000000000 (SMK, KMMK and ALL-PKA only)

Press ENTER to process.
```

Figure 4-30 Resetting the previous PKA Master Key values

The reset function must also be confirmed by pressing Enter (see Figure 4-31).

```
----- ICSF - Restart Key Entry Process -----

ARE YOU SURE YOU WISH TO RESTART THE KEY ENTRY PROCESS?

Restarting the process will clear the ALL-PKA master key register.

WARNING: Resetting the KMMK or SMK will invalidate any private
         internal key tokens in the PKDS
```

Figure 4-31 Confirmation of previous PKA Master Keys reset

We entered PKA Master Keys in two parts because PCICC will require that at least two key parts are needed, first and final (whereas CCF, although not recommended, can have only one PKA master Key part).

To create a new random PKA Master Key part, we used option **5, UTILITY** on the ICSF Primary menu panel. On the ICSF Utilities panel, we first generated the PKA Master Key and calculated checksum for the entry process. This value was entered using Clear Master Key Entry to all those processors we earlier cleared.

Entering the first key part is shown in Figure 4-32.

```
----- ICSF - Clear Master Key Entry -----
COMMAND ==>

          CCF DES/PCICC SYM-MK new master key register   :  EMPTY
          CCF Signature/PCICC ASYM-MK master key register :  EMPTY
          CCF Key Management master key register         :  EMPTY

Specify information below

Key Type  ==> ALL-PKA           (DES, SMK, KMMK, ALL-PKA)

Part      ==> FIRST            (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 6C

Key Value ==> 529B459D89024A5E
          ==> 9BE6EA2F3D62B5CD
          ==> 6D85CDF745C15B40 (SMK, KMMK and ALL-PKA only)

Press ENTER to process.
```

Figure 4-32 Entering the first PKA Key part

Entering the final key part is shown in Figure 4-33 on page 137.

```
----- ICSF - Clear Master Key Entry -----
COMMAND ==>

          CCF DES/PCICC SYM-MK new master key register   : EMPTY
          CCF Signature/PCICC ASYM-MK master key register : PART FULL
          CCF Key Management master key register          : PART FULL

Specify information below

Key Type ==> ALL-PKA          (DES, SMK, KMMK, ALL-PKA)

Part      ==> FINAL          (RESET, FIRST, MIDDLE, FINAL)

Checksum ==> DE

Key Value ==> 4F3D01D9F1BCFE5E
          ==> 62CE982A38138AC7
          ==> 837C5B62DC04498F (SMK, KMMK and ALL-PKA only)

Press ENTER to process.
```

Figure 4-33 Entering the final PKA Master Key part

After all PKA master Key parts have been entered, the PKDS will be enciphered by selecting the **Master Key** option from the ICSF Primary menu pane, and then option **5, Reencipher PKDS** (see Figure 4-34 on page 138).

**Reminder:** PKDS reencipherment requires at least one PCICC to be installed and enabled in the system.

```
----- ICSF - Master Key Management -----  
  
Enter the number of the desired option.  
  
 1 INIT/REFRESH CKDS - Initialize a Cryptographic Key Data Set or  
                        activate an updated Cryptographic Key Data Set  
 2 SET MK              - Set a DES master key  
 3 REENCIPHER CKDS    - Reencipher the CKDS prior to changing the DES  
                        master key  
 4 CHANGE MK          - Change the DES master key and activate the  
                        reenciphered CKDS  
  
 5 REENCIPHER PKDS    - Reencipher the PKA Cryptographic Key Data Set  
 6 ACTIVATE PKDS      - Activate the PKDS after it has been reenciphered  
 7 REFRESH CACHE      - Refresh the PKDS Cache if enabled  
  
Press ENTER to go to the selected option.  
Press END  to exit to the previous menu.  
  
OPTION ==> 5
```

Figure 4-34 Select the Reencipher PKDS

On the REENCIPHER PKDS panel, we gave the name of the current PKDS as input PKDS and the name of the new preallocated PKDS as output PKDS (see Figure 4-35 on page 139), and pressed Enter.

```
----- ICSF - Reencipher PKDS -----  
  
To reencipher all PKDS entries from encryption under the old signature/  
asymmetric-keys master key to encryption under the current master key  
enter the PKDS names below.  
  
Input  PKDS ==> 'PATKAP.PKDS4'  
  
Output PKDS ==> 'PATKAP.PKDS5'  
  
  
Press ENTER to reencipher the PKDS.  
Press END  to exit to the previous menu.
```

Figure 4-35 PKDS reencipherment

After a successful PKDS reencipher, the message REENCIPHER SUCCESSFUL will be shown (see Figure 4-36).

```
----- ICSF - Reencipher PKDS ---- REENCIPHER SUCCESSFUL  
  
To reencipher all PKDS entries from encryption under the old signature/  
asymmetric-keys master key to encryption under the current master key  
enter the PKDS names below.  
  
Input  PKDS ==> 'PATKAP.PKDS4'  
  
Output PKDS ==> 'PATKAP.PKDS5'
```

Figure 4-36 PKDS reenciphers successfully

After reencipherment, the new PKDS has to be activated. Select option **6**, **ACTIVATE PKDS** on the Master Key Management panel; the Activate PKA Cryptographic Key Data Set panel will be shown (see Figure 4-37 on page 140). Press Enter to activate the new PKDS.

```

----- ICSF - Activate PKA Cryptographic Key Data Set -----
Enter the name of the new PKDS below.

New PKDS ==> 'PATKAP.PKDS5'

```

Figure 4-37 Activating the new PKDS

The final step is to enable PKA Callable Services through Administrative Control Functions (see Figure 4-38). Remember to also enable PKDS Read Access and PKDS Write, Create and Delete Access, because simply enabling PKA Callable Services does not enable those services.

```

----- ICSF - Administrative Control Functions -- Row 1 to 4 of 4

Active CKDS: PATKAP.CKDS4
Active PKDS: PATKAP.PKDS5

To change the status of a control, enter the appropriate character
(E - ENABLE, D - DISABLE) and press ENTER.

FUNCTION                                STATUS
-----                                -
. Dynamic CKDS Access                   ENABLED
e PKA Callable Services                  DISABLED
e PKDS Read Access                       DISABLED
e PKDS Write, Create, and Delete Access  DISABLED
***** Bottom of data *****

```

Figure 4-38 Enabling PKA Services

After enabling all those PKA services, the system log will contain the messages shown in Figure 4-39 on page 141.

```
CSFM116I BOTH MASTER KEYS CORRECT ON PCI CRYPTOGRAPHIC COPROCESSOR P01,  
SERIAL NUMBER 92E01983.  
***
```

Figure 4-39 PCICC card becomes active after enabling PKDS services

The status of the PKA services will change to ENABLED (see Figure 4-40).

```
----- ICSF - Administrative Control Fun          FUNCTION CHANGED  
  
Active CKDS: PATKAP.CKDS4  
Active PKDS: PATKAP.PKDS5  
  
To change the status of a control, enter the appropriate character  
(E - ENABLE, D - DISABLE) and press ENTER.  
  
FUNCTION                                STATUS  
-----                                -  
. Dynamic CKDS Access                    ENABLED  
. PKA Callable Services                  ENABLED  
. PKDS Read Access                       ENABLED  
. PKDS Write, Create, and Delete Access  ENABLED  
***** Bottom of data *****
```

Figure 4-40 PKA Services enabled

### 4.4.7 UDX-related definitions in the OPTIONS Data Set

Before UDX can be used, you need to make changes to the Options Data Set.

ICSF will recognize the new UDX service through the UDX card (refer to Figure 4-41 on page 142), where UDX-id (“XX”, in our case) is supplied to the installation when the UDX is developed.

The service-number specifies a number that identifies the service to ICSF. Valid service numbers are 1 to 32767, inclusive. We used “50” as our service number.

This set of service numbers is valid for both installation-defined services and UDX services. The service number of a UDX service must not be the same as the service number of an installation-defined service.

The load-module-name (“UDXSRV”, in our case) is the name of the module that contains this service. During ICSF startup, ICSF loads this module and binds it to the service-number that was specified.

A comment may be specified. The positional parameter is required. The comment consists of up to 40 EBCDIC characters, and may include imbedded blank characters. The comment text is enclosed by single quotes.

FAIL option YES specifies that ICSF ends abnormally if the service cannot be loaded.

```
CKDSN(PATKAP.CKDS4)
PKDSN(PATKAP.PKDS5)
DOMAIN(1)
COMPAT(NO)
SSM(YES)
KEYAUTH(YES)
CHECKAUTH(YES)
TRACEENTRY(1000)
USERPARM(USERPARM)
COMPENC(DES)
REASONCODES(ICSF)
PKDSCACHE(0)
UDX(XX,50,UDXSRV,'TEST UDX FOR CRYPTO RESIDENCY',FAIL(YES))
EXIT(CSFPCI,UDXEXIT,FAIL(NONE))
```

*Figure 4-41 Options Data Set with UDX-related definitions*

The EXIT option should be used together with the Trusted Key Entry. This exit (“UDXEXIT”, in our case), gets control during PCI Interface callable service. The exit will verify that the usage of the UDX code is enabled from the TKE Workstation.

If there is no TKE installed in the system, all ICSF callable services are allowed for the execution.

**Tip:** On the 4758 Web page ([www.ibm.com/security/cryptocards](http://www.ibm.com/security/cryptocards)), select **library** (on the left side) and a list of documentation will be displayed. Use the instructions detailed in “zSeries User Defined Extensions Reference and Guide” for guidance on how to create UDX.

## 4.4.8 Installation of the UDX shown in ICSF panels

Information about the newly installed UDX can be displayed using ICSF panels by first selecting **9, UDX MGMT**, from the ICSF Primary menu panel. The panel shown in Figure 4-42 on page 143 will appear.

```
----- ICSF - User Defined Extension Management -----  
  
Enter the number of the desired option.  
  
 1 Display authorized UDXs for a coprocessor  
 2 Display the coprocessors where a UDX is authorized  
 3 Authorize a UDX  
  
Press ENTER to go to the selected option.  
Press END  to exit to the previous menu.  
  
OPTION ===> 1
```

Figure 4-42 UDX management main panel

If you need to find out which UDXs are installed in the system, select option **1** on the UDX management panel. On the panel (see Figure 4-43 on page 144), select the Coprocessor you wish to be queried.

**Note:** The query can be targeted only on the PCI Crypto Cards that appear on the Coprocessor list with serial numbers. The ones without serial numbers are PCI Crypto Accelerator cards.

```

----- ICSF - Authorized UDX Coprocessor Selection Row 1 to 4 of 4

Select the coprocessor to be queried and press ENTER.

      COPROCESSOR      SERIAL NUMBER      STATUS
      -----
/ P00                  92E01846          ACTIVE
. P01                  92E01983          ACTIVE
. P02                                ACTIVE
. P03                                ACTIVE
***** Bottom of data *****

```

Figure 4-43 Selecting Coprocessor for the UDX query

The result of the query is shown in Figure 4-44.

```

----- ICSF - Authorized UDXs ----- Row 1 to 1 of 1

For PCI Cryptographic Coprocessor P00

UDX id  Service Module  Comment
-----
XX      UDXHRV          TEST UDX FOR CRYPTO RESIDENCY
***** Bottom of data *****

```

Figure 4-44 Display the UDX information installed into selected PCICC

If you need to determine on which PCI Crypto Coprocessor a certain UDX is installed in the system, select option 2 on the UDX management panel. On the panel (see Figure 4-45 on page 145), give the UDX id that you wish to be queried.

```
----- ICSF - Coprocessors for Authorized UDX -----  
  
Enter the two character id of the User Defined Extension to be queried.  
  
UDX id ==> XX
```

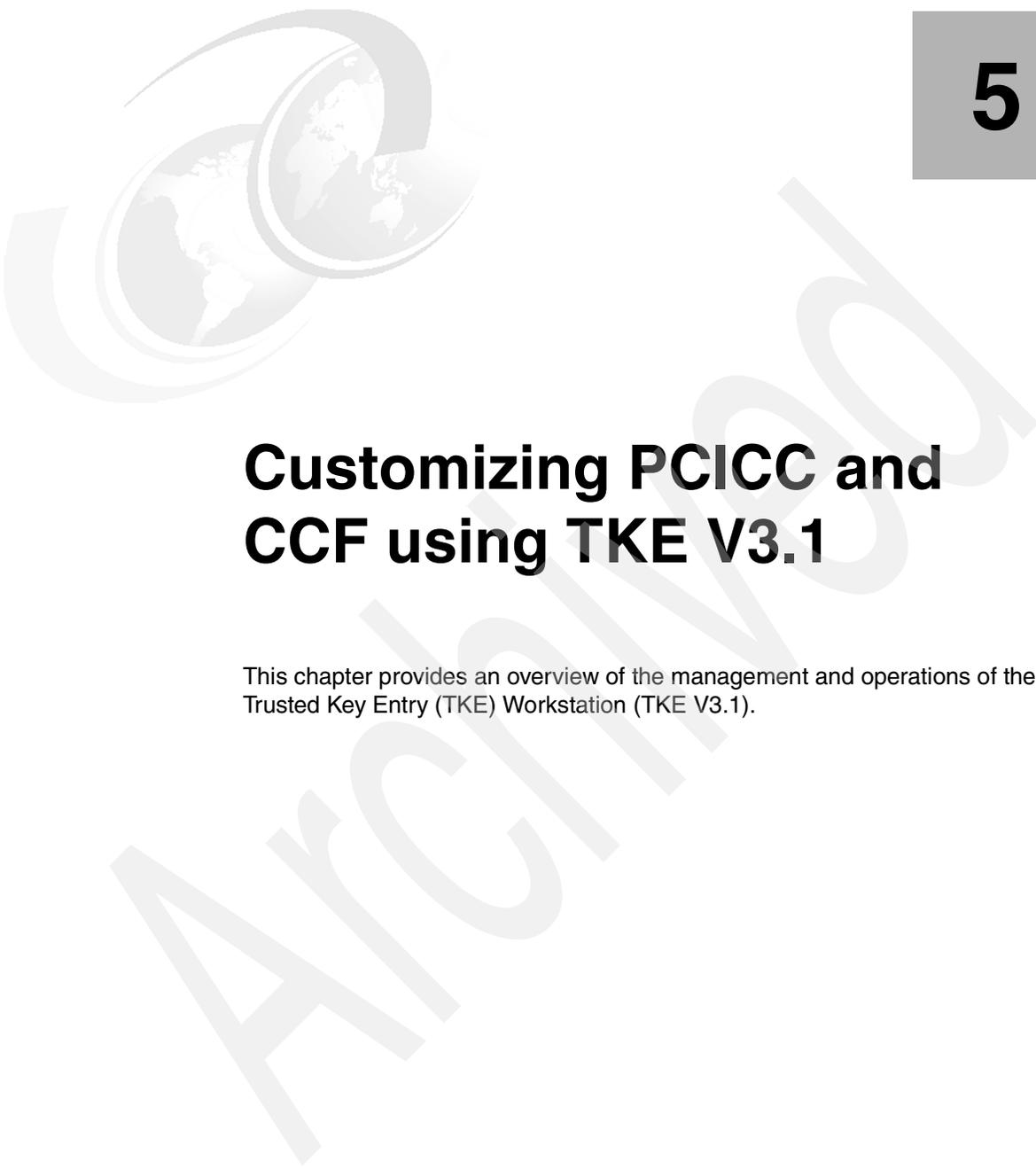
Figure 4-45 Locate the Coprocessor where UDX is installed

The result of the query is shown in Figure 4-46.

```
----- ICSF - Coprocessors for Authorized UDX -- Row 1 to 1 of 1  
  
User Defined Extension XX is authorized on the following coprocessors:  
  
  COPROCESSOR      SERIAL NUMBER      STATUS  
  -----  
  P00              92E01846          ACTIVE  
***** Bottom of data *****
```

Figure 4-46 Show the Coprocessor where the UDX is installed

Archived



## Customizing PCICC and CCF using TKE V3.1

This chapter provides an overview of the management and operations of the Trusted Key Entry (TKE) Workstation (TKE V3.1).

## 5.1 Introduction to the TKE V3.1 Workstation

With the introduction of the PCICC card for 9672 G5/G6 processors, the TKE Workstation has been enhanced to allow remote administration of the PCICC card(s). It also continues to support the standard Cryptographic Coprocessor Feature.

Security and secrecy are provided by encrypted and signed transactions between the TKE and the remote Crypto coprocessors. Diffie-Hellman generated keys are used to protect the key parts transported between the TKE and the S/390 host cryptographic coprocessors, and requests and responses flowing over the network are authenticated using public key cryptography.

The graphical interface has also been revisited, so users should find the module management panels and menus of the TKE V3.1 application friendlier to use. More details can be found in the *TKE Workstation User's Guide 2000*, SA22-7524.

### 5.1.1 Major changes

The following are brief descriptions of the major changes to this workstation.

- ▶ The cryptographic functions in the TKE V3.1 are performed by an IBM 4758 model 002 cryptographic coprocessor installed in the workstation (the TKE V2.x uses a 4755 cryptographic adapter).
- ▶ The TKE V3.1 can manage both PCICCS and CCFs. (The TKE V2.x supports only the CCF modules, and there is no upgrade path between TKE V2 and TKE V3)
- ▶ Figure 5-1 on page 149 illustrates the compatibility between the TKE versions and models of the 9672 and zSeries servers.

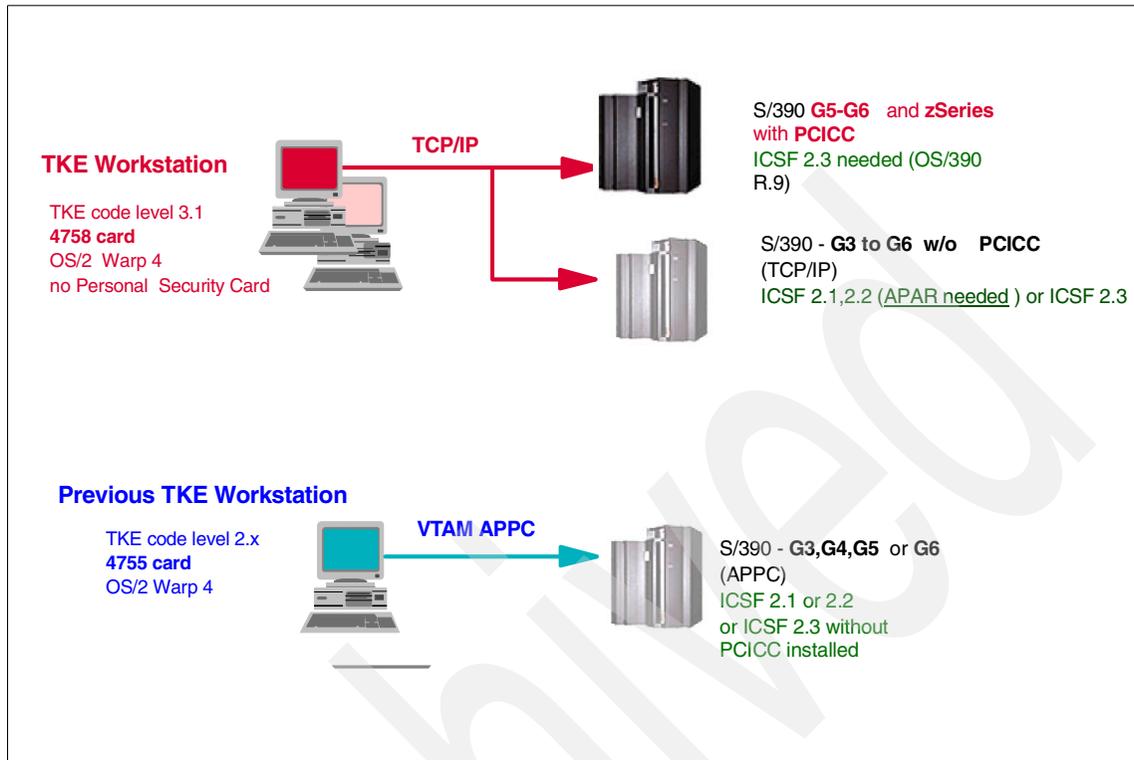


Figure 5-1 TKE Workstation configuration according to CMOS models

- ▶ The TKE Workstation V3.1 communicates with the host system using the TCP/IP network via a token ring or ethernet adapter card, as shown in Figure 5-2 on page 150.

The TKE z/OS host has started, on top of TCP/IP and ICSF, a TKE Host Transaction Program which listens over a dedicated TCP/IP port and serves the requests issued by the connected TKE(s).

**Note:** The TKE option feature must be ordered with the CCF enablement feature code, as there is no PCICC FCV enablement feature code that includes TKE support.

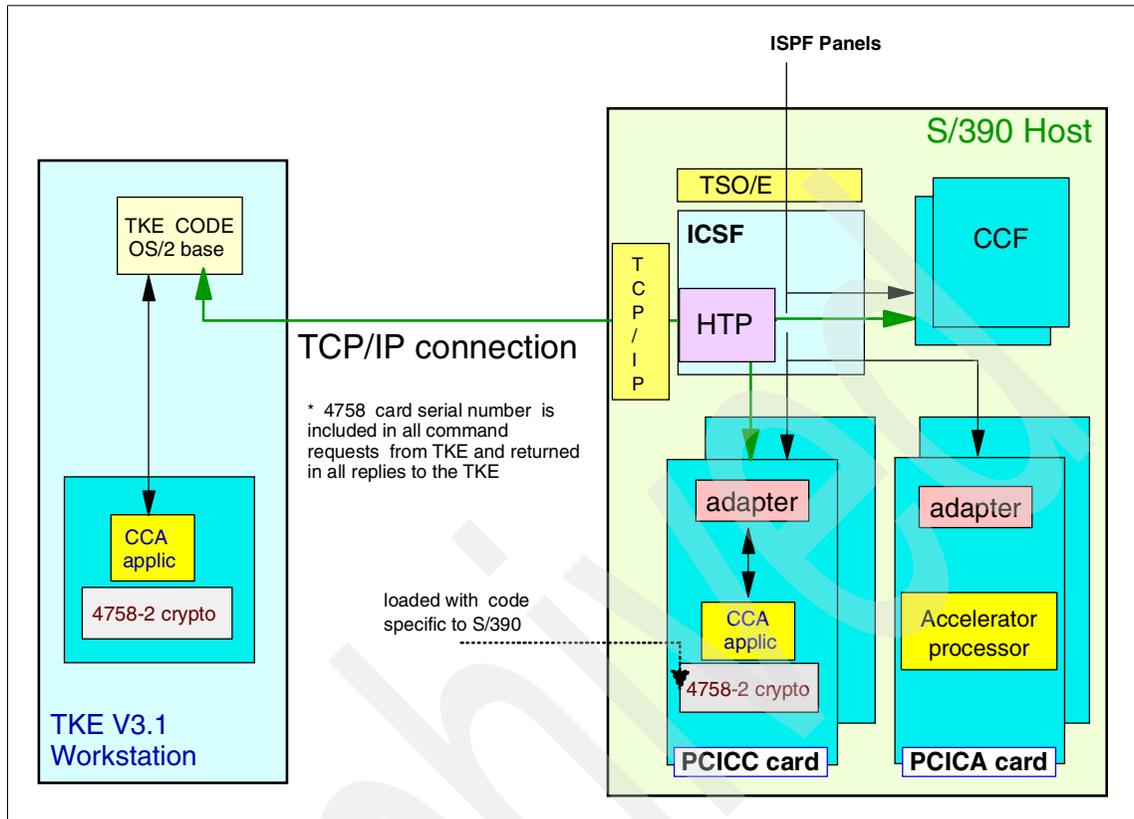


Figure 5-2 TKE Workstation in PCI CC and CCF environment

► TKE customization

The TKE Workstation is shipped from manufacturing with code already loaded and parameters set in default values.

- Step one will define TKE authorities (administrator, keymen), set the TKE Master Key, and define TKEUSERS profiles (people who will manage host cryptographic processors). TCP/IP connection parameters to the host partition must also be entered.; see Figure 5-3 on page 151.
- Step two will define and customize host names, Host CCFs/PCICCs authorities, and access control parameters; see Figure 5-4 on page 152.

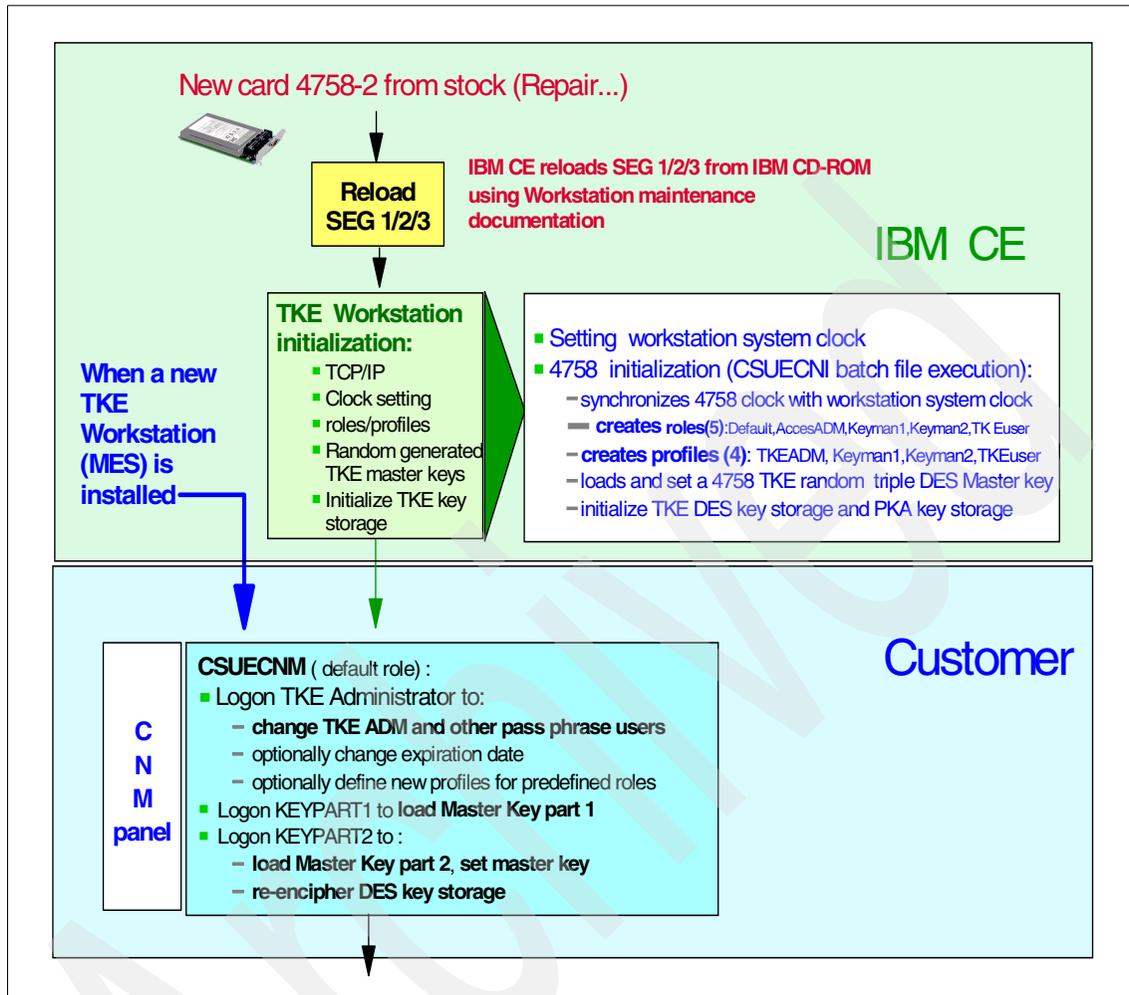


Figure 5-3 TKE V3.1 internal card 4758 setup process data flow - part 1

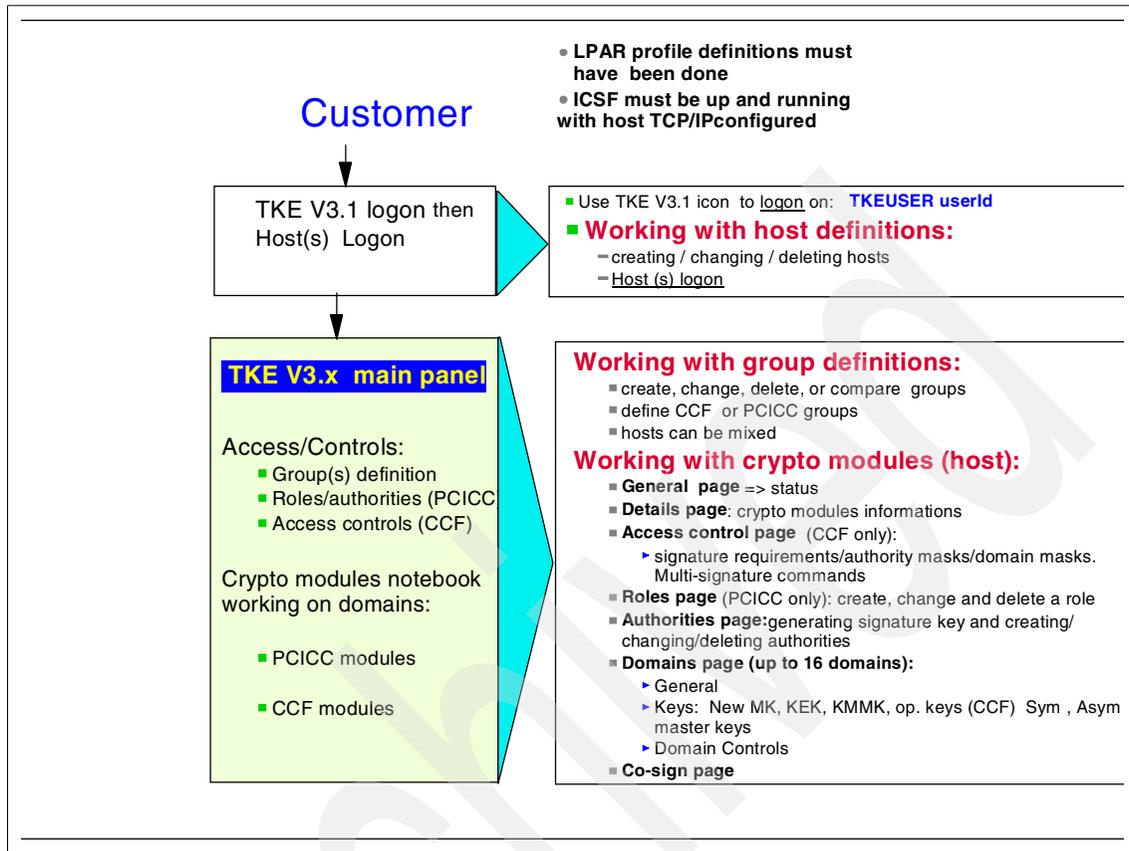


Figure 5-4 TKE process data flow - part 2: host definitions, host CCFs, and PCICC customization

### 5.1.2 Before using the new TKE

Some initialization and configuration steps must be performed *before* using the new TKE Workstation. These are described in detail in *TKE Workstation User's Guide 2000*, SA22-7524, Appendix A.

Some preliminary setups are also required:

1. The customer security policy should be defined and roles assigned: the TKE security officer; the TKE Master Key officer(s); the TKE user(s); the Host Crypto modules security officer(s); the Host Master Keys officer(s), and so on.
2. The logical partitions' cryptographic coprocessor definitions should be done, and the proper LPARs should be activated.

3. The TKE z/OS host should have TCP/IP, ICSF, and the TKE Host Transaction Program properly configured and started. Refer to 5.3, “TKE application: managing host Crypto coprocessors” on page 170 for more information.

### 5.1.3 The TKE V3.1 software

The software installed on the TKE V3.1 includes the following:

- OS/2 Warp 4.0
- IBM4758 PCI cryptographic coprocessor CCA support program 2.20 for OS/2

This code is loaded in the 4758 card at the TKE Workstation manufacturing location. This code controls the secure cryptographic functions performed inside the 4758 secure hardware. The code integrity and security is protected by the IBM private key signature when loaded into the card. Any further reload or update of this code will be integrity-guaranteed by the same mechanism.

- Trusted Key Entry Version 3.1 (versus 2.x)
- Java runtime environment 1.1.8

### 5.1.4 TKE Workstation installation - general information

During the manufacturing process, the TKE Workstation application is loaded in the PC (currently, an IBM 6792) hard disk. TKE IBM4758 mod 002 card internal CCA code segments 1, 2, 3, and FCV are also loaded in the cryptographic card.

The TKE is then tested. Just before shipment, the cryptographic coprocessor card IBM 4758 is removed from the workstation and shipped in a special thermal-protected container. When received at the customer location, it is re-plugged in the TKE Workstation by an IBM representative.

**Note:** When manipulating the 4758 card, *never* remove the batteries; otherwise, the tamper protection mechanism will definitively disable the card.

The following are shipped along with the workstation:

- ▶ A CD-ROM containing the TKE code
- ▶ A TKE backup diskette
- ▶ A TKE binary diskette
- ▶ *TKE Workstation User's Guide 2000, SA22-7524*

The TKE Workstation application can be reloaded from the TKE CD-ROM and the previously saved backup diskette.

**Note:** If for any reason the IBM 4758 Crypto card has to be replaced, then the internal CCA code segments 1, 2, and 3 have to be reloaded because new spare 4758 cards only have the minimum code bootstrap loaded at the card manufacturing plant. It will also be necessary to reload the FCV. You can accomplish this using the procedure described in *Maintenance Information for Desktop Consoles*, GC38-3115, to load the code files that are kept on the workstation hard disk.

### 5.1.5 TKE definitions

In this section, we explain TKE terminology.

#### **TKE Workstation access control**

TKE 4758 cryptographic card access control mechanisms use the *roles and profiles* concept (refer to Figure 5-5 on page 155). When necessary, these roles and profiles are defined using the Crypto Node Management (CNM) software facility.

This is done under the control of a TKE administrator and according to customer security policy. This facility, included in TKE code, is started from an OS/2 window session as shown in Figure 5-11 on page 163.

#### **ROLE**

A role defines a class of TKE users. When creating or changing a role, the TKE administrator will define the TKE 4758 Crypto card commands that will be authorized for users mapped to this role. Each command the card can perform can be individually enabled or disabled according to the setting of an “Access Control Point” assigned to that command. See the IBM4758 CCA code support documentation for details on the control access points.

TKE has a DEFAULT role. Use of the DEFAULT role does not require a user profile. Any user can use the services permitted by the DEFAULT role without logging onto or being authenticated by the coprocessor. However, the role is limited in the operations it is allowed to perform.

**Note:** Predefined roles are set up during TKE initialization, and there is no need to change these roles for normal use of the TKE application, as shown in Figure 5-10 on page 162.

#### **PROFILE**

A profile defines a specific user. It contains a user name and pass phrase information and is mapped to one role, and only one role. However, as many profiles as needed can be created and mapped to the same role.

After logging on under his/her profile, the TKE user will be able to perform only the commands authorized by the role mapped to the profile. Validity dates may also be defined on a role basis to enforce specific security policies.

Default profiles are set up during TKE initialization and have a default known pass phrase. We recommend that you change this pass phrase after TKE setup to insure full access security.

The TKE administrator will have to create all user profiles required by the security policy in effect; see Figure 5-13 on page 165.

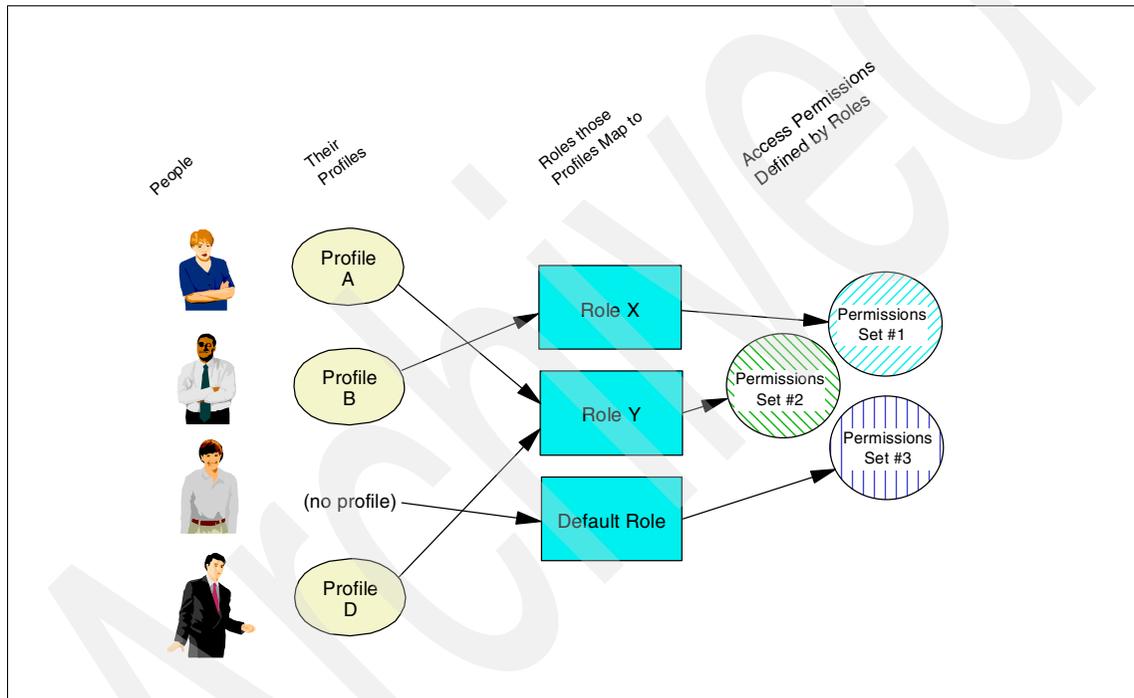


Figure 5-5 Role-based access control

**Important:** The PCICC cards and CCF chips in the host system use a very similar access control concept, as they also use roles and authorities. However, these roles and authorities apply to the S/390 coprocessors and must not be confused with TKE 4758 roles and profiles.

To summarize:

- ▶ The TKE Workstation has its own ROLES and USER PROFILES.
- ▶ The PCICC modules have their own ROLES and AUTHORITIES.
- ▶ The CCF modules have their own AUTHORITIES.

### ***TKE Master Key***

The TKE IBM4758 card contains one Master Key (referred to as the DES Master Key) to encrypt the TKE operational DES keys, and one Master Key (referred to as the PKA Master Key) to encrypt the PKA keys to be stored on TKE hard disk in files DESSTORE.DAT and PKASTORE.DAT (also named TKE keystores). The CNM utility is used to set these two Master Keys.

You enter the DES Master Key in several key parts loaded by different key officers (generally, two officers) through the CNM utility. The entered key parts can be imported from a file, manually keyed, or randomly generated.

Once the last Master Key part is loaded, the officer sets the Master Key via the CNM utility, which causes both the DES Master Key and the PKA Master Key to be set from these key part values in the TKE 4758. Then the officer uses CNM to encipher or re-encipher the key storages.

### ***TKE Workstation key storages***

The TKE Workstation key storages are files on hard disk where the operational keys encrypted under the TKE IBM 4758 Master Key are stored. When these keys need to be used, they are loaded into the 4758 secure hardware, so that they never appear in clear outside of the TKE 4758 secure hardware.

There are two key storages implemented in the TKE:

- ▶ DES key storage to store symmetric keys (C:\ibm4758\DESSTORE.DAT)
- ▶ PKA key storage to store asymmetric keys (C:\ibm4758\PKASTORE.DAT)

## **5.2 TKE Workstation TCP/IP setup**

The TKE administrator must configure the workstation for access via a TCP/IP network. To start this configuration, from the TKE OS/2 desktop, double-click the

TCP/IP icon to open the Configuration Notebook, as shown in Figure 5-6 on page 157.

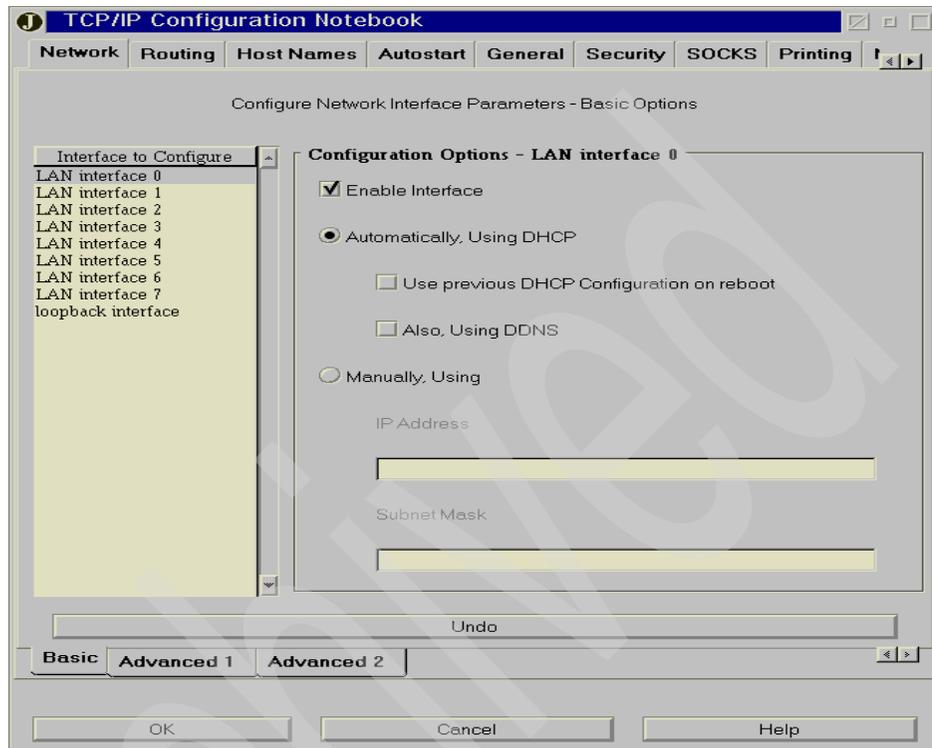


Figure 5-6 TCP/IP Configuration Notebook

- Fill in the basic options page of the Network tab for at least one adapter, typically LAN interface 0.
- In our example, the option Automatically using DHCP was used. This setup is dependent on the specific network implementation, and has to be performed with the assistance of the customer network support personnel.
- Read the Configure Routing Information online help to determine if any special situations apply to the computer you are configuring.

If so, you may need to complete the field on the Routing tab (however, this was not needed in the case of our DHCP network); refer to Figure 5-7 on page 158.

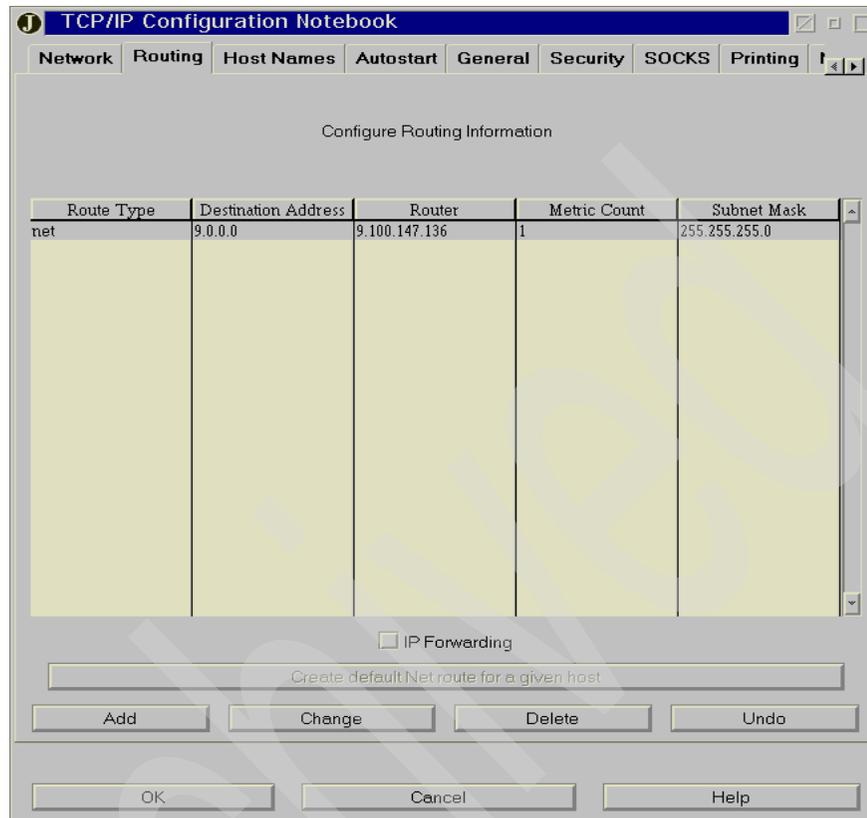


Figure 5-7 TCP/IP routing tab

- Then go to the Autostart tab in the Configure Automatic Starting of Services page. Select **routed** from the Autostarted Services list, check the **Autostart Service** box, and then press **OK**.

We started the routed daemon in our configuration. This is shown in Figure 5-8 on page 159.

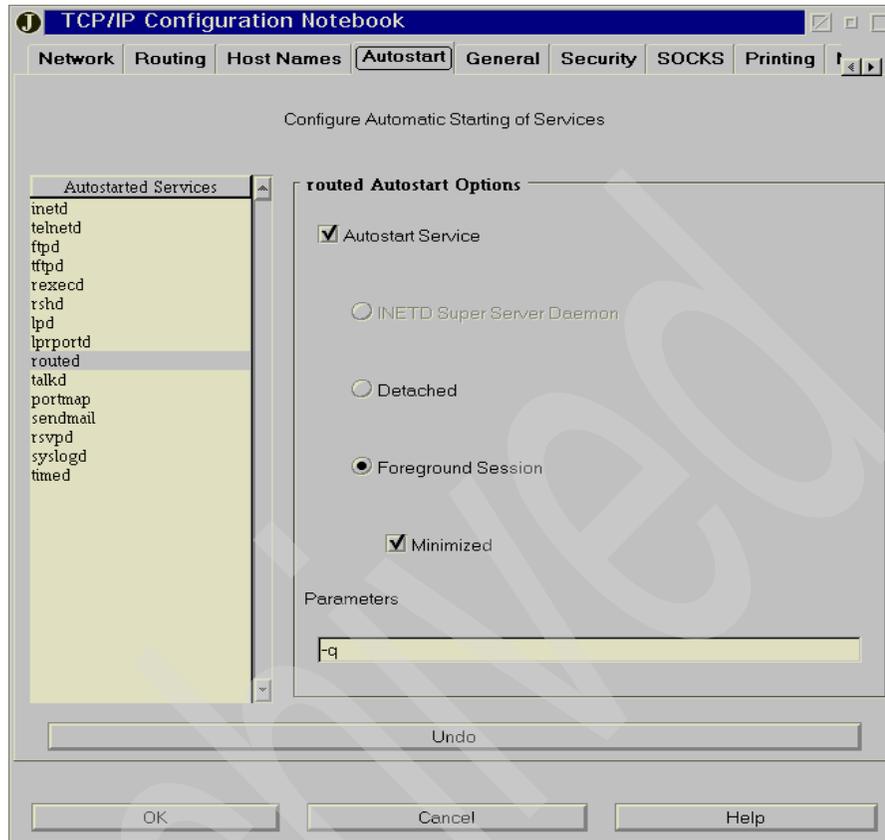


Figure 5-8 Autostart tab

- On the Configure Name Resolution Services pages, select the **Host Names** tab and provide the host name, local domain, and name service addresses as shown in Figure 5-9 on page 160. (This is not needed with a DHCP network.)

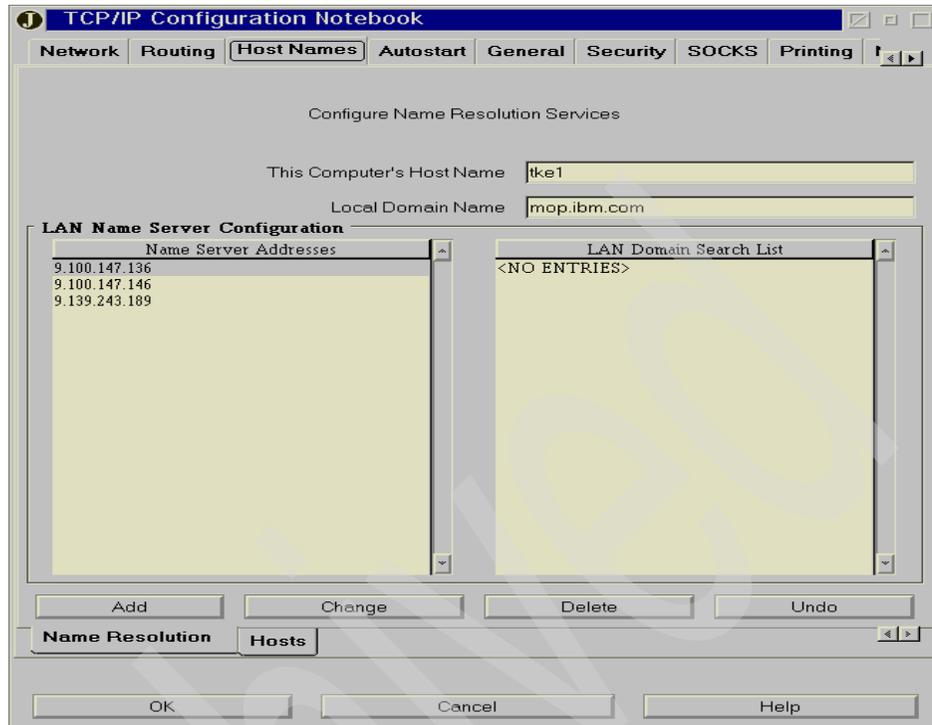


Figure 5-9 Host Names tab

## 5.2.1 z/OS TCP/IP Host Transaction Program

The setup of the Host Transaction Program is explained in Appendix E, “TKE host TCP/IP server setup” on page 331.

## 5.2.2 TKE Workstation 4758 setup

TKE Workstation initialization has to be done only when installing a new TKE or after a TKE misuse, when there is no TKE role or profile available to manage the TKE and no backup diskettes are available.

Normal restoration of a failing TKE must be done using the CD-ROM and the backup diskettes.

**Important:** The workstation you are working on is already in use, and if some keys are already stored in key storages, then ensure that you have a backup copy of *both* key storage files *and* the TKE 4758 card Master Keys parts.

Otherwise, you will not be able to recover the keys in the key storages; refer to 5.3.6, “Backing up the TKE files” on page 238 for more information.

## TKE Workstation initialization

This step is performed at the manufacturing location before the TKE is shipped and does not have to be done again when you receive the TKE. However, it can be re-executed without any problem.

The description of the actual TKE customer.setup begins at 5.2.3, “TKE access control administration” on page 163.

To start TKE Workstation initialization, enter the following from the OS/2 window:

```
c:\ibm4758\cnm\ csuecni c:\tke\4758access\4758initialize.cni
```

This CSUECNI command runs the batch file 4758initialize.cni, which will perform the following:

1. It sets the 4758 clock.
2. It creates five TKE default roles (shown in Figure 5-10 on page 162).
  - The DEFAULT role allows you to view roles and profiles, and to re-initialize the 4758. This is a public access, and no pass phrase is needed (simply press Cancel when prompted for profile name and pass phrase).
  - The TKEADM role is the default administrator role, allowing the user to perform security administration for the TKE Workstation and to create, change, or delete TKE roles and profiles.

**Note:** The TKEADM role may have been called the ACCESADM role in early TKE V3.1 code releases.

- The KEYMAN1 role allows you to clear the TKE 4758 new Master Key reg and to load the first Master Key part.
- The KEYMAN2 role allows you to load the middle and last Master Key parts, to set the Master Key, and to re-encipher the TKE key storage.
- The TKEUSER role allows you to communicate with the host Crypto modules, and to manage them. This role is also used by the 4753

migration facility (see 5.4, “4753 Key Token Migration facility” on page 240 for more information about this facility). This is the TKE general user role.

3. It creates four TKE default profiles, as shown in Figure 5-10 on page 162:
  - The TKEADM profile is mapped to the TKEADM role, with the pass phrase TKEADM.
  - The KEYMAN1 profile is mapped to the KEYMAN1 role, with the pass phrase KEYMAN1.
  - The KEYMAN2 profile is mapped to the KEYMAN2 role, with the pass phrase KEYMAN2.
  - The TKEUSER profile is mapped to the TKEUSER role, with the pass phrase TKEUSER.
4. It loads and sets a TKE IBM 4758 random DES Master Key.
5. It initializes TKE DES key storage and TKE PKA key storage.

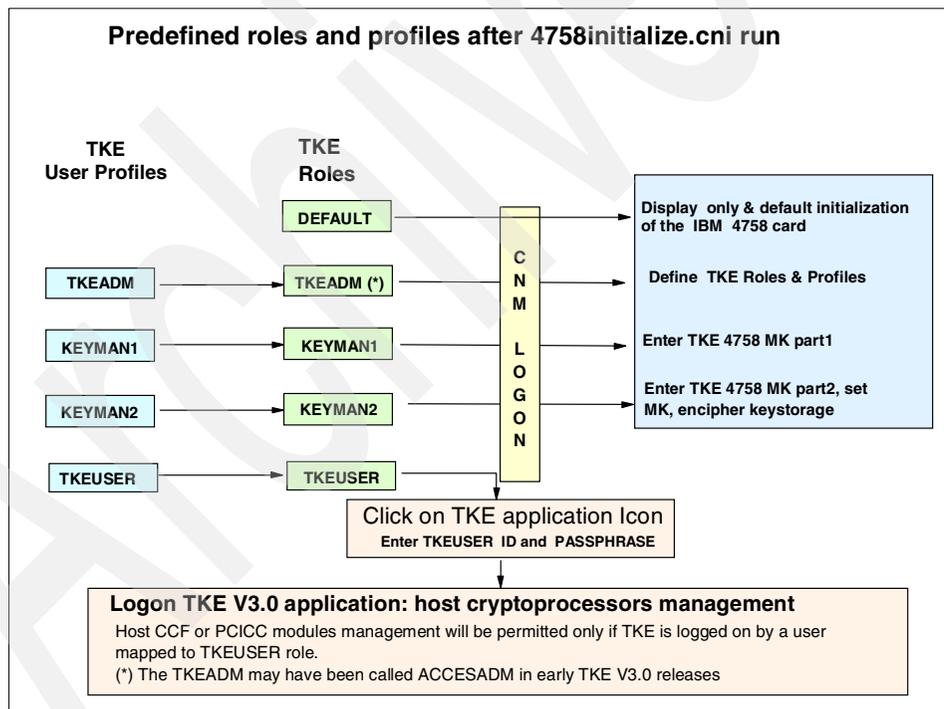


Figure 5-10 TKE Workstation predefined roles and profiles

### 5.2.3 TKE access control administration

This step is used to change the TKE user profiles default pass phrases and replace the IBM4758 Master Key which was randomly generated during the TKE initialization process, as no backup is available to recover this key. These tasks are executed using the Cryptographic Node Management (CNM) utility.

The CNM utility is a basic application delivered with the IBM 4758 cryptographic coprocessor. Only a subset of the CNM functions is required to customize the TKE and to manage the workstation application.

**Note:** We strongly recommend that you do *not* use other CNM functions, as this can lead to unpredictable results. In particular the CNM Initialize function must not be confused with the TKE initialization previously described.

Refer to the IBM 4758 PCI Cryptographic Coprocessor CCA Support Program installation manual for additional details on CNM.

To start the CNM utility, enter the following command from an OS/2 window:

```
C:\ibm4758\cnm\CSUECNM
```

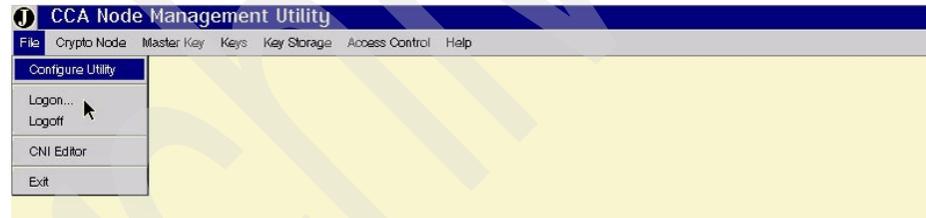


Figure 5-11 TKE Workstation CNM utility panel

The panel shown in Figure 5-11 will be displayed. When the CNM utility is started, only the DEFAULT role commands are permitted. As previously mentioned, the DEFAULT role allows you to only display roles and profiles and to initialize the 4758.

#### **CNM TKE administrator logon**

From the File pull-down menu on the CNM panel, select **logon**. Enter the user ID TKEADM and the pass phrase TKEADM (these are the default values set at TKE initialization), as shown in Figure 5-12 on page 164. Note that the user ID and pass phrase are case sensitive.

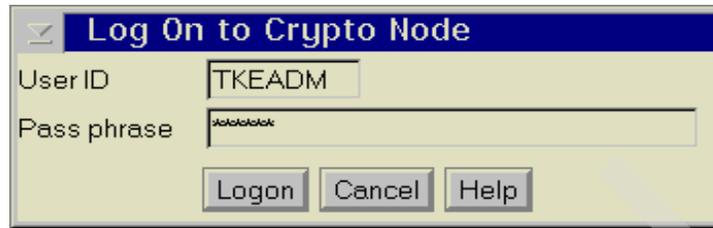


Figure 5-12 CNM utility logon as default TKEADM user with default pass phrase

The panel shown in Figure 5-11 will be displayed. When the CNM utility is started, only the DEFAULT role commands are permitted. As previously mentioned, the DEFAULT role allows you to only display roles and profiles and to initialize the 4758.

Once logged on, the TKE administrator needs to change the pass phrase and may also change the activation and expiration dates for the TKEADM profile. For detailed information on how to edit profiles, refer to the IBM 4758 PCI Crypto Coprocessor CCA Support Program manual.

At this time the administrator should also change the default pass phrases in the other predefined profiles and, if needed, create new user profiles mapped to the predefined roles.

To create or change profiles, select **profiles** in the access control pull-down menu and then press **New** (to create a new user profile) or press **Edit** (to modify a selected existing profile). The administrator must fill in the panel with the information on the new user and the pass phrase (the pass phrase may be typed in by the user). The administrator will also need to define which role is attributed to this user. An example is shown in Figure 5-13 on page 165.

It is expected that most of the new user profiles will be mapped to the predefined TKEUSER role, as this is the only predefined role permitted for communication with the host Crypto modules.

In our example we decided to keep the default TKEUSER user ID and pass phrase. This, of course, may change according to the customer security policy at your site.

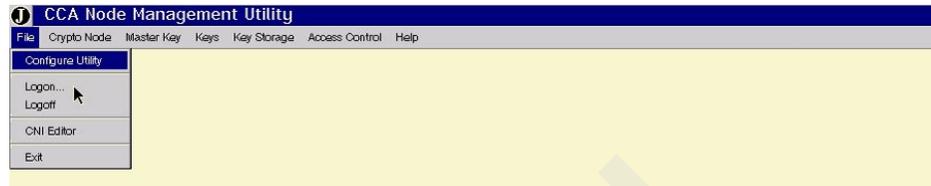


Figure 5-13 TKE Workstation profile example: TKEUSER default profile

To create or change roles, select **roles** in the access control pull-down menu and then press **New** (to create a role) or press **Edit** (to change a selected existing role). An example is shown in Figure 5-14 on page 166.

As explained in 5.1.5, “TKE definitions” on page 154, there is normally no need to change or create roles for the TKE Workstation IBM 4758 card since the required functions are already permitted in the pre-defined roles. However, you can use this as an opportunity to change the validity days specification.

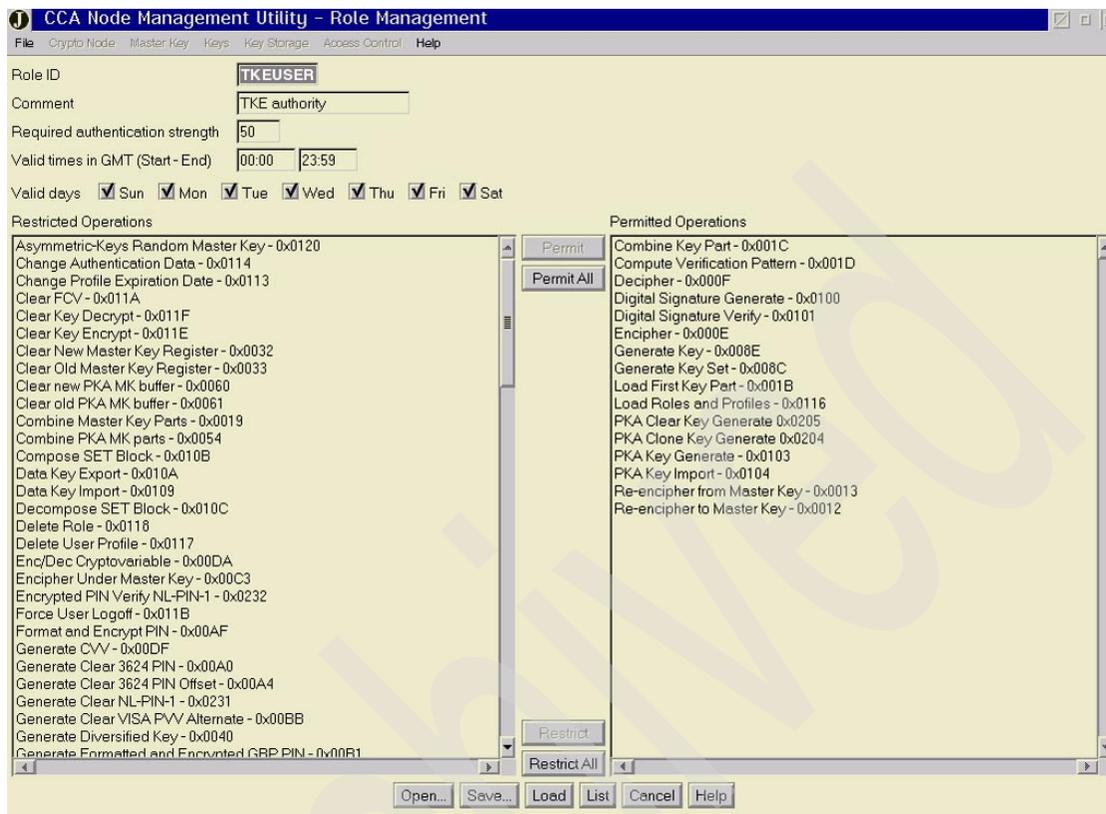


Figure 5-14 TKE Workstation role example: TKEUSER default role

As shown in Figure 5-14, the left column of the role panel shows the unauthorized operations and the right column shows the operations permitted to the role. You can move operations from one column to the other, according to the specific requirements of your site—but keep in mind that any change in the roles default commands setup may induce unexpected results when using TKE to manage host crypto modules. Valid days of the week can be also selected.

When the creation or modification of roles, profiles, and pass phrases is complete, the TKEADM user ID logs off from the Crypto node using the File drop-down menu Logoff option.

### **Loading the first part of the TKE Master Key**

During TKE initialization, the TKE IBM 4758 DES Master Key is loaded using randomly generated key parts. We strongly recommend, for recovery purposes, that you change this Master Key with your own backed-up key parts. This is done

by using KEYMAN1 for the first key part and KEYMAN2 for the middle and last key parts, as follows:

1. Logon to the CNM utility as KEYMAN1.
2. From the Master Key pull-down menu, select **Parts** and then **first key part**. Manually enter the 24 bytes of the first key part, or else select **generate random** key part. Then select **Load**.

**Notes:**

- ▶ The 4758 new Master Key register must be empty before you begin this task. To clear the new Master Key register, select **clear new register** from the Master Key drop-down menu.
- ▶ Ensure that you save the key part for backup purposes. The key part can be saved on paper or saved on diskette by using **Save...**; refer to Figure 5-15 and Figure 5-16 on page 167 for illustrations.

3. KEYMAN1 must then logoff from CNM.



Figure 5-15 TKE Workstation MK - first part generate and load



Figure 5-16 TKE MK - first part successfully loaded

**Loading the last key part of the TKE Master Key**

To load the last key part of the TKE Master Key, do the following:

1. Logon to the CNM utility as KEYMAN2.
2. From the Master Key pull-down menu select **parts**, and then **enter and load a middle key part** (optional) and the **last key part**. Perform a save for each one.

### **Setting the Master Key and ENCIPHER DES and PKA key storage**

To set this key and storages, do the following:

1. Stay logged on as KEYMAN2.
2. From the Master Key pull-down menu, select **Set** to set the IBM 4758 Master Key. A successful completion message must be displayed, as shown in Figure 5-17 on page 168.



Figure 5-17 Set TKE Workstation Master Key successful panel

KEYMAN2 now has to reencipher the DES and PKA key storages. To proceed, do the following:

1. From the key storage pull-down menu, select **DES storage, Manage**, then press **REENCIPHER**.
2. Repeat the operation for PKA storage.

At completion, KEYMAN2 must log off from the Crypto node and then close the CNM utility.

### **Customize tke.ini parameters**

You can customize a number of parameters by editing the tke.ini file; first issue this command:

```
cd tke\tke
```

Then, using E, edit the tke.ini file:

- ▶ **Blind key entry option**  
This option will mask the key parts as they are entered at TKE when working with the host system Crypto modules. If it is required by the security policy, you will have to add the following statement, if it does not already exist:  
**BLIND\_KEY\_ENTRY=TRUE**
- ▶ **TRANSPORT\_KEY\_POLICY** (keys used to protect TKE-to-Host transactions)  
This option selects the default key transport policy. The allowed values are 1, 2, or 3. This value is more conveniently changed from the TKE main application panel function; refer to Figure 5-18.

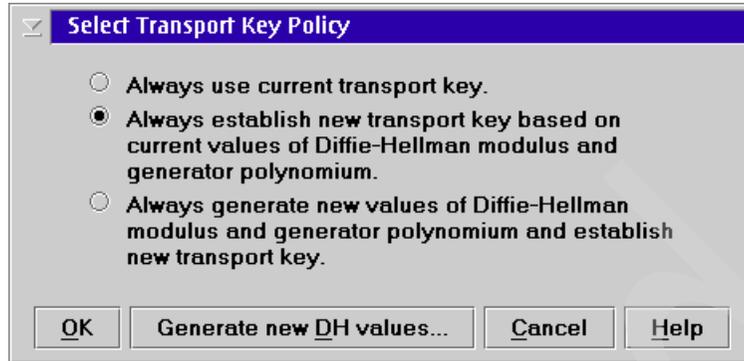


Figure 5-18 Define key policy on main TKE panel

► **DEFAULT DIRECTORY**

This option specifies the directory selected when opening a window to specify where to save an authority signature or a key part.

Initially this is set to default; it will be changed the first time you save a key to a binary file.

► **SERIALIZE\_PATH**

This option specifies where the TKE saves internal informations about hosts and groups. The files HOST.DAT and GROUPS.DAT are saved there.

► **MESSAGE\_PATH**

This option specifies the directory where TKE (on user request) saves the error information in a file named TKE.MSG.

► **If your security policy requires that generated keys be saved to diskette and not to the hard drive, you must enable the FLOPPY\_DRIVE\_ONLY feature.**

This function will forces generated key parts to be saved on floppy diskettes. Make sure FLOPPY\_DRIVE\_ONLY=TRUE.

Save the file and close the OS/2 window.

## 5.2.4 Starting the TKE application

TKE initialization and setup is now complete. You can start the application by opening the TKE folder and double-clicking the TKE V3.1 icon.

## 5.3 TKE application: managing host Crypto coprocessors

At this point it is assumed that the TKE initialization and setup have been performed, and that TCP/IP, ICSF, and the Host Transaction program have been started in the TKE z/OS host system.

The definition of the S/390 coprocessors and their handling at the TKE will be done according to the customer policy. Therefore it is assumed that as a prerequisite to this chapter the user has already defined which users are authorized to: change the Crypto modules setup; change the domain Master Keys; zeroize a domain; enable or disable a module; and so on.

### 5.3.1 Managing modules

When a new PCICC or CCF processor is added, it must be configured in the TKE application. A PCICC card or CCF module can be managed at the TKE as a single PCICC or CCF coprocessor, or can be included in a PCICC or CCF coprocessor group.

Managing the coprocessors as a group provides easier management when dealing with several coprocessors in the installation. Group management introduces the concept of the “master” Crypto coprocessor in the group, which is a single coprocessor that will be representative of the status of all the other coprocessors in the group.

**Note:** Because it is important that *all* Crypto modules in the group are in the same state (that is, they have the same definition of signature requirements for CCF, or roles/authorities for PCICC cards), we recommend that you continue to control at the group level once this has been initiated.

If it happens that a command issued to a group is performed only by some modules in the group, then the TKE application adds two new group definitions: one group contains the Crypto coprocessors where the command has been properly executed, and one group contains the coprocessors which have not properly executed the command. These two new groups can be used to manually resolve the differences in the module states and synchronize them again. Once this is done, these groups can be deleted.

Before adding a new module in an already existing group, the administrator and/or different authorities will have to customize the new Crypto module as a single entity; then, when the status of the coprocessor is identical to the group's master coprocessor, it can be added to the group. Compare function in notebook main panel will compare the module and list the differences, if any.

**Important:** The decision to work either with single coprocessors or with groups of coprocessors should be made at the installation level.

## Required levels of authorization

Working with the S/390 Crypto coprocessors requires the following three levels of authorization (this is illustrated in Figure 5-19 on page 172):

1. All commands initiated on the TKE Workstation that imply communication with host Crypto coprocessors will have to be performed with the TKE application logged on using a user profile mapped to the TKEUSER role (or to a role with equivalent permissions). This is controlled by ROLES and PROFILES defined at the TKE level by the TKE administrator.
2. Access to the host system cryptographic facilities is requested by using a TSO user ID and password known from the host system.
3. The TKE authority sending the command to a specific host Crypto coprocessor must be authorized to perform and sign this command. This is defined at the host system Crypto coprocessors for each authority index by the authority administrator.
  - A just-installed PCICC card has a pre-set default authority mapped to a default role (INITADM). This authority will be used until new roles and authorities have been defined for PCICC administration.

An *authority* is identified by its index and has its own signature key pair. Authority 00 is the pre-set default authority for the PCICC. At the TKE, the authority 00 default signature key must be loaded (from the TKE main panel functions pull-down menu) so that commands can be accepted by a newly installed (or zeroized) PCICC.

This initial default authority 00 is mapped to a predefined INITADM role in the PCICC, which is authorized to create new roles and authorities in the PCICC card.

- A just-installed CCF module has 16 preset authorities. At the TKE, the default signature must be loaded (from the TKE main panel functions pull-down menu) so that commands can be accepted by a newly installed (or zeroized) CCF.

There is no way to create or delete CCF authority. Initially, the same default signature is assigned to authorities 0 to 13. Authorities 14 and 15 have their own default signature.

**Note:** Authorities 14 and 15 cannot be used for signing commands until their default private and public keys have been changed for new keys.

All these CCF authorities default signatures have to be updated and access control-customized according to the installation's security policy.

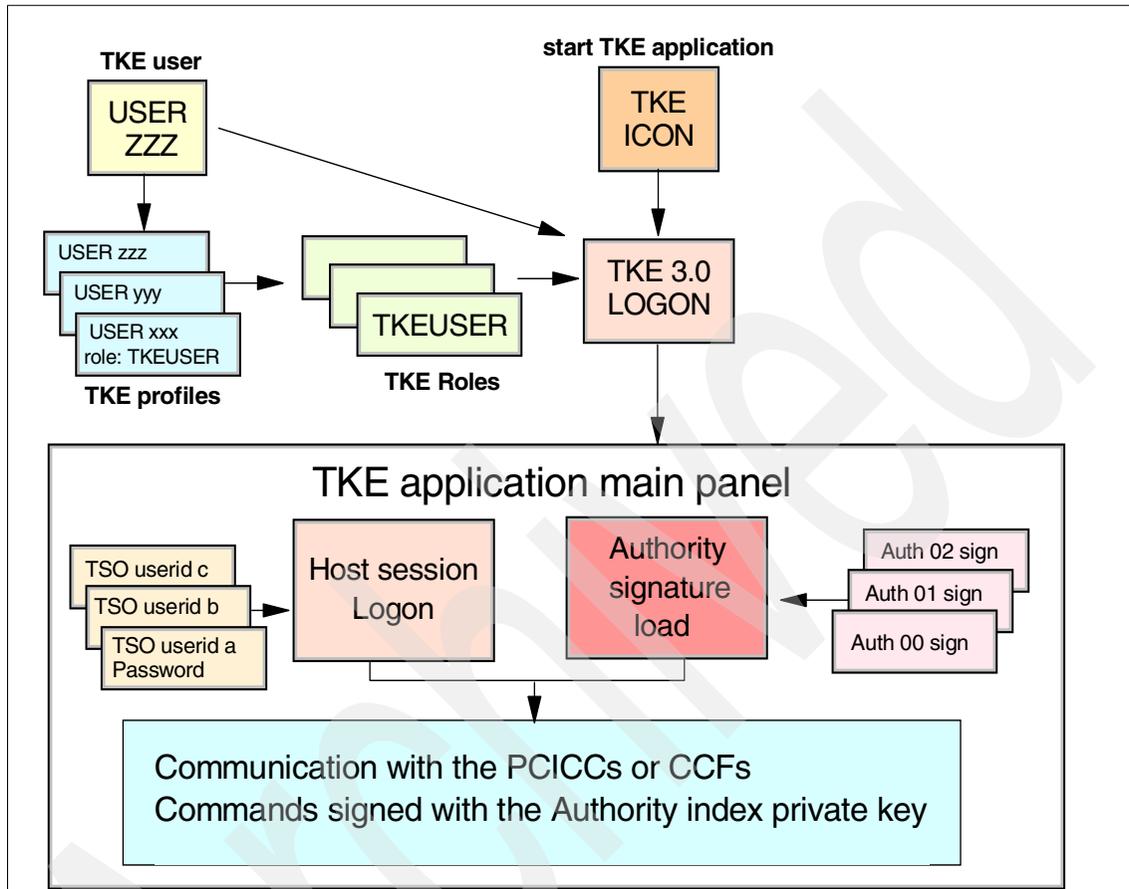


Figure 5-19 Host Crypto module access logon sequence from TKE

### **Authority signature key indicator**

An informational message indicating: authority signature key loaded or not loaded is displayed at the lower right-hand corner of the TKE window. This indicates whether the TKE can sign commands sent to host coprocessors.

Only one signature key can be loaded at a time. It can be changed only from the TKE main panel, and the signature key source can be a binary file, the key storage, or the default signature key.

If no signature key is loaded, and if an authority attempts to send a command to any host Crypto module, a pop-up message will appear, requiring a signature key source to be entered.

**Note:** If a signature key is already loaded, there is no information about its owner. This signature key will be used to sign any further command. Hence, before sending commands, *be sure* that the loaded signature key is the one you intended. If in doubt, the safest action is to return to TKE entry main panel and then load your signature key again.

### 5.3.2 PCICC and CCF setup on the TKE Workstation

The following sections provide details on how to perform these setups.

#### TKE logon

The TKE must be logged on using a profile mapped to the TKEUSER role. If the TKE application is already active, close the application and restart it by clicking the **Trusted Key Entry** folder and then the **TKE V3.1** icon. Enter the user ID and pass phrase (remember, the characters are case sensitive) as shown in Figure 5-20. As already mentioned, we kept the default TKEUSER profile.



Figure 5-20 TKE Workstation logon panel

The main TKE panel is displayed. Initially this panel is empty, as shown in Figure 5-21 on page 174.

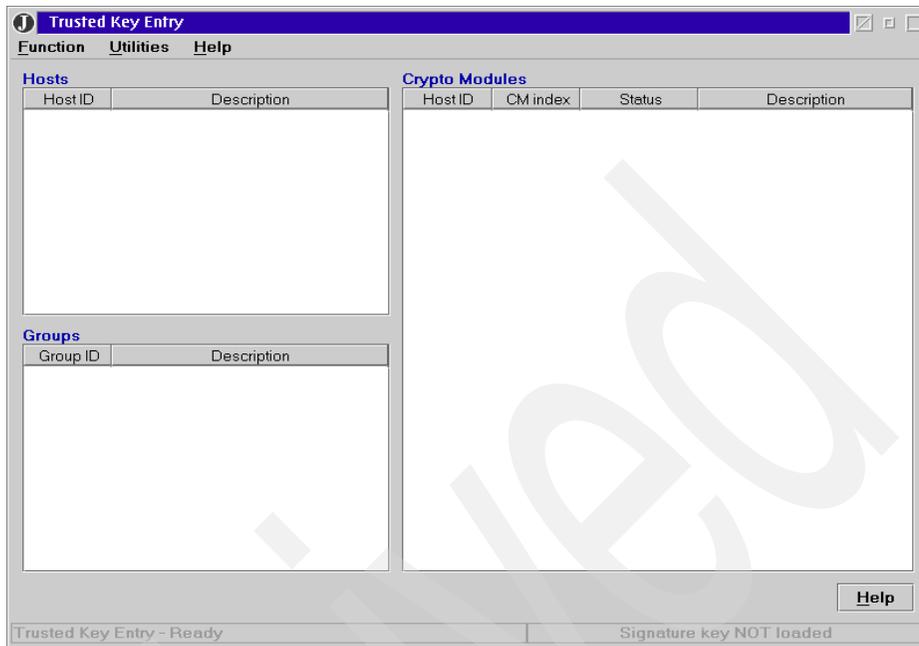


Figure 5-21 TKE V3.1 application main panel at first logon

Use this main panel to log on to the host system, then select a Crypto module or a group of the same type (PCICC or CCF) of Crypto modules. As a prerequisite, you must enter host and group definitions in the empty panels. (Figure 5-21 displays the informational message: Signature key NOT loaded in the lower right-hand corner.)

The main panel has three panels: Hosts, Groups, and Crypto Modules. These panels are blank until a host is created, modules are authenticated, and groups are defined.

This panel also contains three pull-down menus: Function, Utilities, and Help. We explain the Function and Utilities menus in more detail in the following sections.

### **Function pull-down menu**

The Function pull-down menu allows you to:

- ▶ Load a signature key by selecting the **Load signature key** in this menu, then selecting the key source, as shown in Figure 5-22.
- ▶ When dealing with a *new* PCICC or CCF coprocessor, you must select **Default key** for authority index 00, as these are the values preset in the coprocessor.

(If working with modules already customized, you will have to select **B**inary file to import the signature of the desired authority index.)

Only one authority index signature can be loaded, so loading the signature must be done each time authority dealing with the host crypto is changed.



Figure 5-22 Load signature key; Source: default signature key

- ▶ Define a transport key policy by selecting the **Define transport key policy** option.

This defines the policy that the user wants to enforce when generating the TKE and host coprocessor transport key by the Diffie-Hellman algorithm. A pop-up menu proposes three policy options; refer to *TKE Workstation User's Guide 2000*, SA22-7524, for further details about these options.

- ▶ Exit.

### **Utilities pull-down menu**

The Utilities pull-down menu allows you to manage TKE DES key storage and PKA key storage. From this panel, you can delete existing keys saved in DES or PKA keystore; see Figure 5-23 on page 176. You may refer to *TKE Workstation User's Guide 2000* for detailed information.

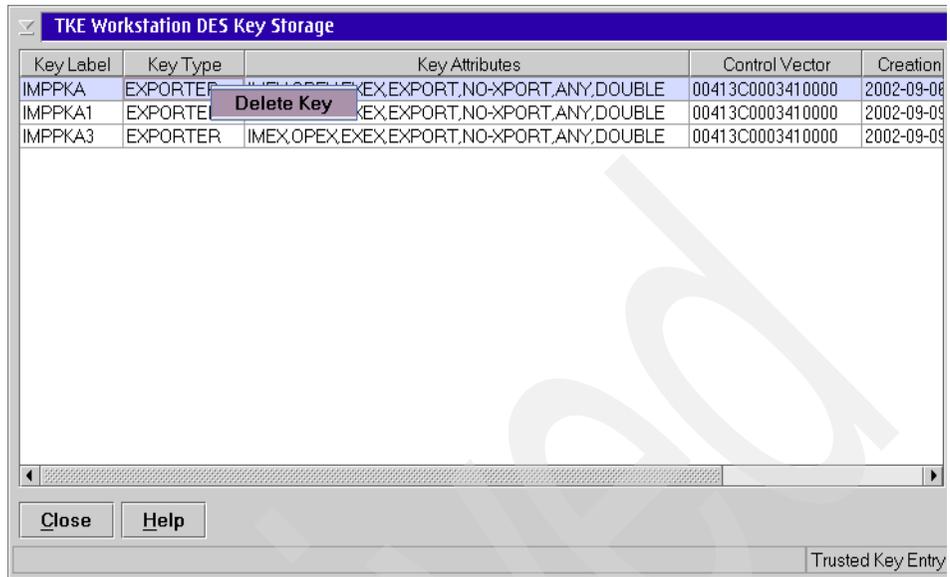


Figure 5-23 Keystore management: Delete Key option

## Host system S/390 definition and logon from TKE

In this section we describe how to create a host. Initially, the host container is empty. To create a host, perform the following tasks:

1. Create a host ID by right-clicking in the TKE main panel host panel and selecting **create**.
2. Enter the Host ID, Host description, Host TCP/IP Address, and the Port number used by the Host Transaction Program. Only the TCP/IP address and port must correspond to actual definitions in the z/OS host, as shown in Figure 5-25 on page 177.
3. Pressing **OK** creates a line for this host in the host panel.

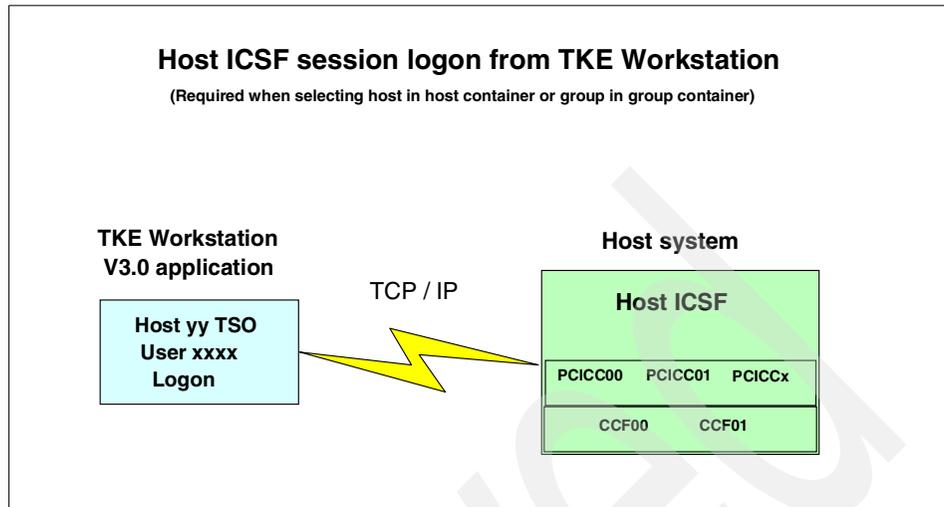


Figure 5-24 TSO Host user ID logon from TKE Workstation



Figure 5-25 Create a new host panel on TKE Workstation - example: NTPREP

To log on to the host, do the following:

1. Click the target host ID line in the hosts panel. The dialog box shown in Figure 5-26 on page 178 appears.
2. Fill in the fields Host user ID and Password with a TSO user ID and password defined in the target host.

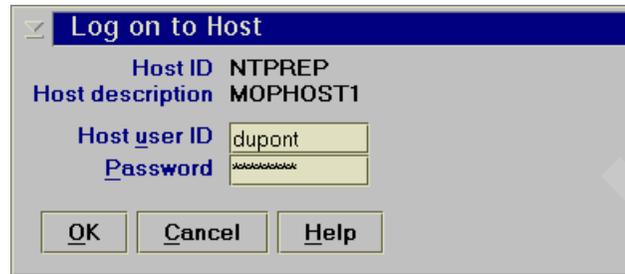


Figure 5-26 Host logon panel on TKE Workstation

Because further processing at the z/OS will be performed under this user ID, we recommend that you consider setting the appropriate RACF profiles as described in 6.1, “RACF access control to ICSF services” on page 242.

### Automated Crypto module recognition

The host PCICCs and CCFs are automatically recognized by the TKE at the time it establishes the first connection with the host. If you added new PCICCs or CCFs and the host is already connected, then exit the TKE application, log on using a TKEUSER profile, and then perform a logon to the target host by clicking **select host** on the main TKE panel host panel to take into account any new Crypto coprocessor installed in the machine.

The TKE user is notified of any new Crypto coprocessor discovered in the host and then needs to authenticate this module CCF or PCICC CMID. The Crypto Module Identifier (CMID) is a serial number burned into secure hardware during the manufacturing process, and it is used to identify the module or the card as a unique one. The CMID is displayed at the TKE and must be verified with information from PCICC or CCF configuration panel on the support element.

After the Crypto module is accepted by the user as the right one, the CMID and Crypto Module Public Modulus (CMPM) are saved in the TKE Workstation for the following communications. The module authentication is illustrated in Figure 5-27 on page 179 and Figure 5-28 on page 180.

**Note:** Contrary to the CCF, the PCICC module identification does not include verification of the CMPM, as these values are provided signed with the IBM private key and verified by the TKE code.

A Crypto module is recognized when its status is “Configured”.



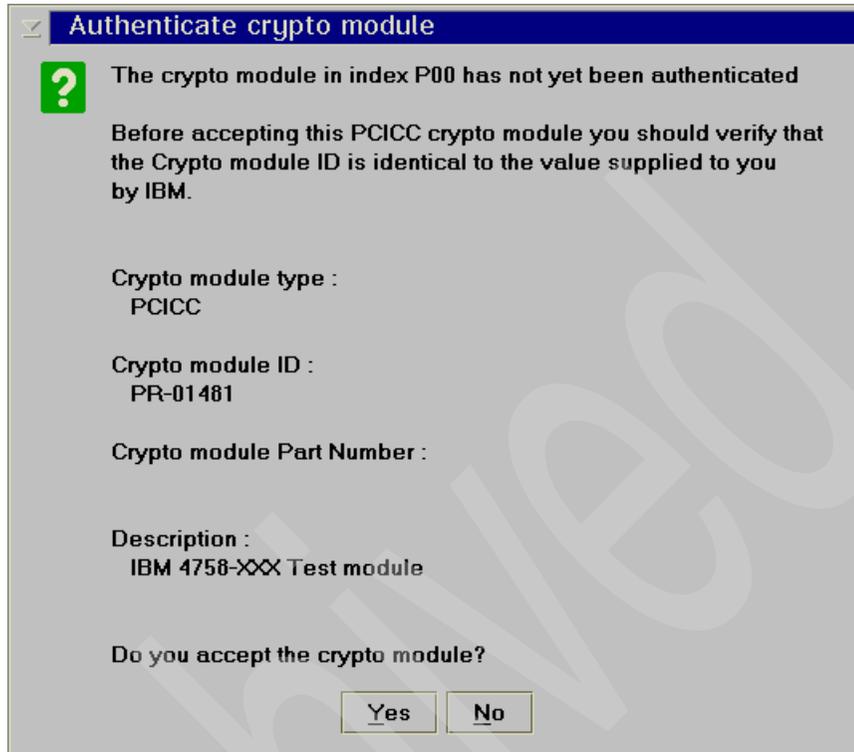


Figure 5-28 PCICC module authentication panel

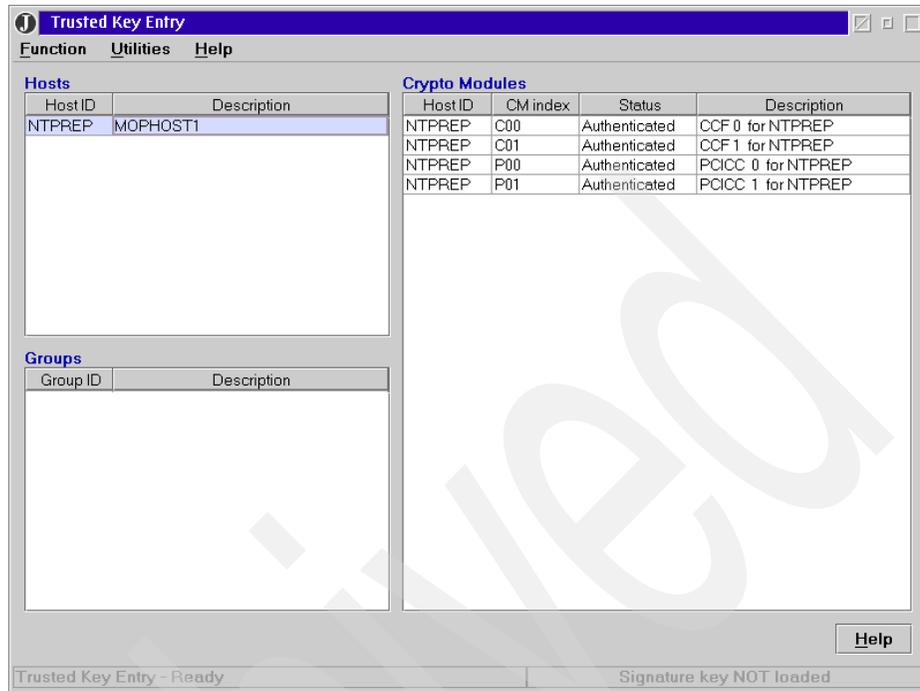


Figure 5-29 TKE main panel after module authentication

### Create or change a group

If you have chosen to work with coprocessor groups, you will have to create a new group or change an already existing CCF or PCICC group. With the mouse right button, click in the Group panel and select **NEW** or **CHANGE**.

Enter (or modify) the group name and group information to specify the module type, select **Host**, and then **Add** (or **Remove**) coprocessors in the group. This is illustrated in Figure 5-30 on page 182.

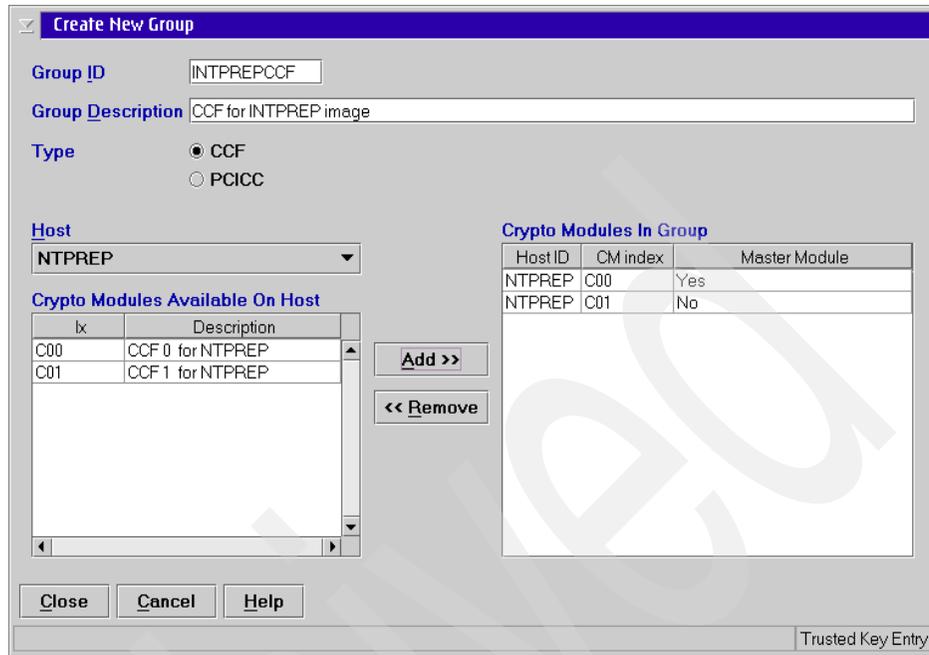


Figure 5-30 Create New Group panel: CCF group consisting of C00 and C01

To choose the coprocessors to be part of a group, select a host in the host list. If not connected to the host, you will be requested to log on. Once connected to the host, this host's coprocessors are displayed and they can be added to the group.

A master Crypto module in the group is also defined; by default, it is the first module added to the group, but it can also be any one of the group's modules. The master module can be changed by right-clicking any module in the group container, and then selecting **master** in the pull-down menu. The states of the Crypto processors can be compared within a group by the administrator.

You can compare the Crypto modules in a group in order to ensure they are all in an identical state by clicking a specific group in the group container, and then clicking any module in the module container.

When the Crypto module notebook is opened, click **Functions** and select **Compare Group**. The TKE reads and compares information from all modules in the group and displays any mismatch it finds. Based on these results, the user can correct mismatches or change group definitions and contents.

### 5.3.3 Manage and update the Crypto module notebook on TKE

The Crypto module notebook is opened by clicking a module in the Crypto panel. Figure 5-35 on page 192 shows the first page of a PCICC module notebook. Further information and controls are accessed by selecting the tabs of the notebook. Figure 5-57 on page 215 shows the CCF notebook general page.

Starting with a new host cryptographic module, you will have to customize authorities and signature requirement or Roles (PCICC), as shown in Figure 5-31.

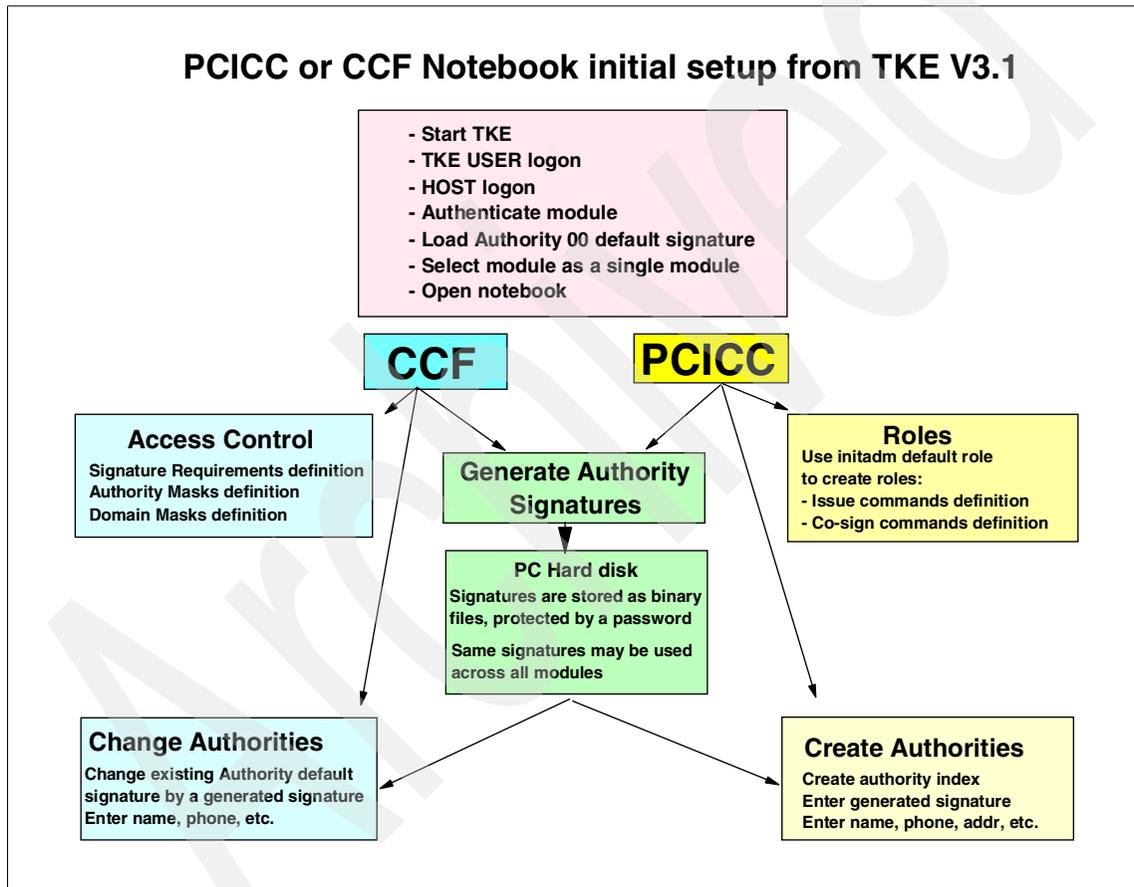


Figure 5-31 PCICC and CCF modules initial setup

At this point, the customer must define the different authorities which will be authorized to manage the crypto modules according to its own security policy. Signatures will be created for each authority and sent to host cryptographic

processors. Control access (CCF) and Roles (PCICC) will be customized for each authority.

Here we have kept the TKE Workstation four default roles and profiles, and created five TKE user profiles mapped to the TKEUSER default role (Paul, Mike, John, Tom, Bill).

These five TKE users have been defined as authorities 01, 02, 03, 04, and 05 for PCICC and CCF modules. The default TKEUSER has been defined as authority 00, using the default signature assigned to the INITADM role (PCICC) and LAP/LCB command (CCF) to prevent any locked situations during training. (Note that this authority will have to be changed or removed during real use of TKE.)

Figure 5-32 on page 185 contains tables that establish a parallel between the multi-signature commands in the CCF and the PCICC functions. Using these tables, you can determine how to maintain consistency between the authorities defined in the CCF and the ones defined in the PCICC.

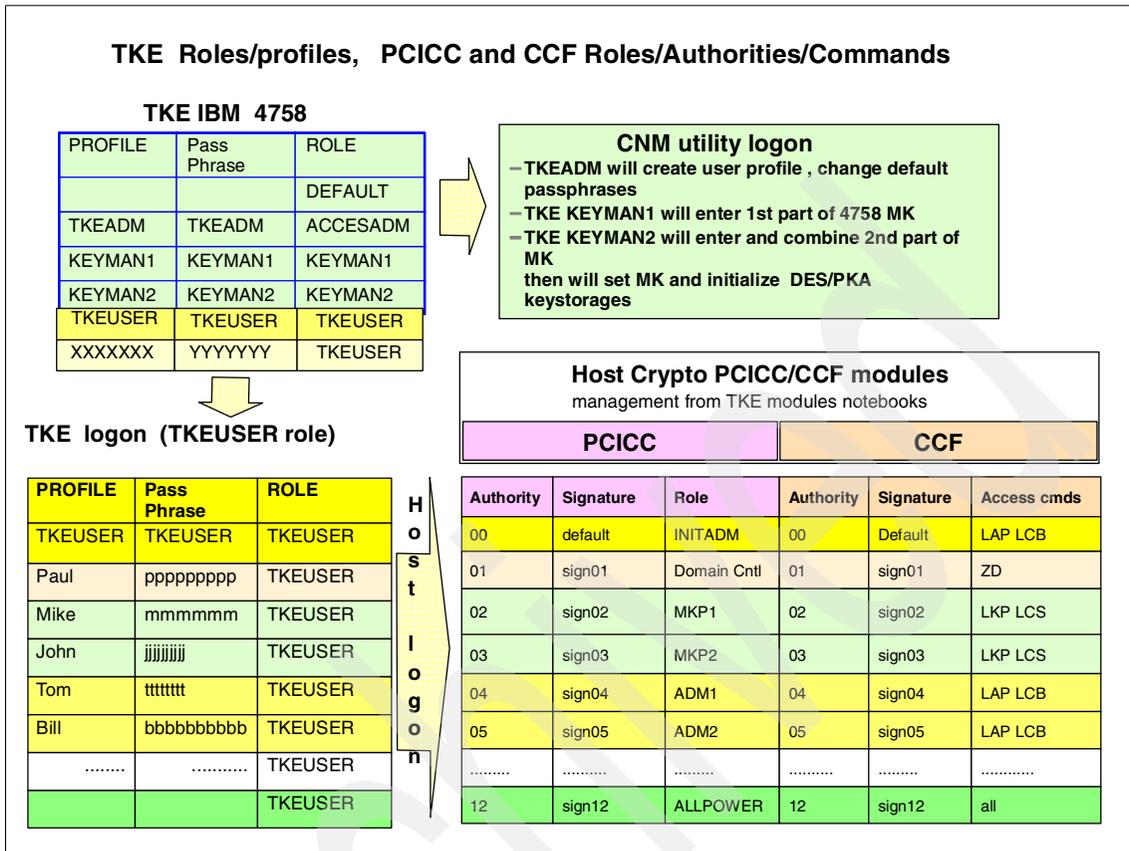


Figure 5-32 TKE roles/profiles, PCICC/CCF roles, authorities and commands

### Crypto module notebook considerations

Working with a Crypto module notebook implies that the TKE Workstation has been logged on by a TKE user ID whose profile is mapped to the TKEUSER role, a host session has been established, and a valid PCICC or CCF Authority signature key has been loaded in the TKE.

The Crypto module notebook allows access to information related to a Crypto module (CCF or PCICC): the CMID, all the authorized commands, the authorities public modulus, the LPAR domain controls, and so on.

### Updating a group of modules

If working with a newly created group made of empty or zeroized modules, select the group from the group container and then click any Crypto module in the

module container to open the Crypto module notebook (only Crypto modules that belong to the group are listed in the module container).

The notebook will internally refer to the group's master coprocessor, but all actions initiated via the notebook will be propagated to all the modules in the group. The Notebook main panel top line will indicate whether the notebook is working with a group, or with an individual module.

## Updating a single module

To access a single module (as opposed to a group), select the host in the host panel and then select the desired Crypto module in the Crypto modules panel. The difference with the group access is that you went through the host selection instead of a group selection.



Figure 5-33 Notebook: working with a PCICC group; P00 is master

Once you have selected a Crypto module or group of Crypto modules, the Crypto module notebook opens on the general page. The Crypto module information displayed is related to the master Crypto module in the group, or to a single selected module. The title line of the notebook will state if you are dealing with a group or an individual coprocessor.

### **Notebook modes**

Notebook opens in one of the following modes: update mode, read only mode, pending command mode, or locked read only mode.

- ▶ update mode

In this mode, you have exclusive control of the module or group of modules. Other TKE users subsequently opening the notebook will get access in read-only mode.

- ▶ read only mode

In this mode, the notebook is already opened by another TKE user. The other user's exclusive control can be released by selecting **release** in the Function pull-down menu.

**Note:** Be aware, though, that release, because it is designed to exit from an unrecoverable lockup situation, may seriously interfere with the work of another TKE with an open notebook for the module.

- ▶ pending command mode

In this mode, the module or group this notebook pertains to has recorded that there is a pending command waiting to be co-signed.

- ▶ locked read only mode

The notebook for a group opens in this mode if one or more modules in the group cannot be accessed from the TKE.

The notebook mode is indicated in the lower right-hand corner of the notebook window.

### **Notebook functions pull- down menu**

- ▶ Refresh notebook: refresh will read the information from the host again.
- ▶ Change signature index: This option will allow you to change the authority index associated to the currently loaded signature. This can be used only if an authority uses the same signature on different hosts, but has a different authority index on each host.

If the signature of the new index is different, you must return to the TKE application main panel function to load the right signature for this index.

- Release crypto module: If the module is locked by another user, this allows you to release the module. Be careful, though, because releasing a module can damage an ongoing operation initiated by another authority.
- Compare: when working with a group, this will compare the parameters of all modules in the group.

## PCICC and CCF Crypto module notebook pages

### **General**

General displays only the coprocessor type and index and the host name, as shown in Figure 5-35 on page 192 and Figure 5-57 on page 215.

### **Details**

Details shows five pages of information for the CCF, and two pages for the PCICC. No changes are allowed in any of these pages.

### **Access control**

Access control (CCF only) are the definitions of the signature requirements for each signed command, with the relevant masks. The contents are unchanged from the previous TKE V2; only the presentation differs. Note, however, that key extract and transfer is no longer available with TKE V3.1.

### **Roles**

Roles (PCICC only): The PCICC uses *role-based* access control. From the TKE Workstation, an administrator defines in the PCICC a set of roles which correspond to classes of TKE users (also known as *authorities*).

Each role defines a set of commands to be permitted. Each TKE authority has a profile in the PCICC, designated by an authority index, mapped to one of the defined roles.

The authority can perform only the commands permitted by the role, and these commands have to be signed by the authority signature key at the TKE. PCICC Roles and profiles are further explained in “Roles and profiles structure example” on page 198.

Figure 5-37 on page 195 shows the commands accessible to the roles in the PCICC.

**Important:** The manipulation of roles in the PCICC can only be executed from the TKE Workstation. Any command performed from the ICSF ISPF panel, or any cryptographic service requested by an application, will be executed under the pre-set DEFAULT role in the PCICC. This DEFAULT role is not accessible for setup.

### **Authorities**

Authorities is a common function used to define authorities and to manage the key pair associated with an authority. Private and public key parts will be stored in TKE key storage and/or stored on media (diskette or TKE hard disk) protected by a password. Note that only one signature key can reside at one time in the key storage.

Before sending or signing a command to a host Crypto module, the authority has to load its signature key in the TKE. This page also allows you to send the public modulus of the authorities to the Crypto modules. This is a prerequisite before sending any signed command to the Crypto modules (except for the default authority key, for which the public modulus is initially known by the Crypto modules).

- ▶ The *CCF* has a fixed number of sixteen authority indices (00 to 15) in effect. We recommend you use the same indices consistently across all coprocessors in the installation.
- ▶ The *PCICC* can record up to 100 authority indices (00 to 99). The number of authority indices in effect is not fixed; it varies with the administrator interventions to add or delete authorities. We recommend, for consistency, with CCF that you only use indices 00 to 15.
- ▶ *Domains*: This entry defines the LPAR domains that can have DES and PKA Master Keys (CCF and PCICCC notebooks) and operational keys (CCF notebook only) loaded and changed from the TKEs. It also provides domain controls.

**Note:** For each of the 16 domains, there are three pages that can be selected with the tabs at the bottom of the window: the domain general page, the domain keys page, and the domain controls page.

For each of the 16 domains, there are three pages that can be selected with the tabs at the bottom of the window: the domain general page, the domain keys page, and the domain controls page.

- ▶ *Co-sign*: Pending commands are processed from this page. When a command requiring multiple signatures (up to 16 signatures for the CCF, or up to two signatures for the PCICC) is issued by one authority, this command stays in pending status until all the signature requirements are completed. The co-sign page displays the pending command details and the signature requirement status. The co-signing authority can either sign the command in this page, or delete the pending command.

There is a difference in the co-sign mechanism between CCF and PCICC for multiple signature commands:

- All authorities defined in the CCF signature requirements panel can independently issue or co-sign a command. There is no restriction on which authorities must be the ones requesting the command first in sequence, and which authorities can subsequently co-sign the issued command.
- Unlike the CCF authority, a PCICC authority is mapped to a PCICC role as being the issuer of a command *or* as the co-signer of a command. Note

that there can be only *one* co-signer to an issued command; while several authorities could co-sign the command, only one co-sign of the command is required.

However, a PCICC role can be given both issuer and co-signer privileges. When an authority mapped to such a role issues a command, the command is executed at once since it is implicitly co-signed.

Figure 5-34 on page 191 establishes a parallel between the multi-signature commands in the CCF and the PCICC functions. Using this table, you can determine how to maintain consistency between the authorities defined in the CCF and the ones defined in the PCICC.

For example, the LAP and LCB commands on CCF and the Access Control Role on PCICC may have the same authorities index because of the similarities of the functions. The LKP (CCF) and SYM MK (PCICC) may also have the same authorities index.

CCF	PCICC
<b>LAP</b> Load Authority public modulus: -Issued when sending an authority signature key from Authority tab	<b>Access control</b> Issued when creating, changing or deleting an authority from Authorities Tab.
<b>LCB</b> load PKSC control block: Issued for setting of signature requirement, authority mask, and domain mask from Access Control tab	<b>Access control</b> Issued when creating, changing or deleting a role from Roles Tab.
<b>ZD</b> Zeroize domain : issued when requesting the domain to be zeroized from Domain General tab	<b>Domain Zeroize</b> -Issued when zeroizing a domain -Issued when co-sign Zeroize From Domain General tab
<b>LEC</b> Load environment control mask : issued when updating crypto capabilities from Domain control tab	<b>Domain Control</b> Load domain controls From Domain control tab
<b>LKP</b> Load Key part:s issued from the load and load to queue functions from Domain keys tab	<b>SYM MK</b> Load Master Key part: 1 Load Master Key final part From Domain keys tab
<b>LCS</b> Load and Combine PKA master keys: issued from the load and reset functions from Domain keys tab	<b>ASYM MK</b> Load and Combine PKA master keys: issued from the load and reset functions from Domain keys tab
	<b>Enable/Disable module</b> From module general tab

Figure 5-34 CCF and PCICC command families: equivalent CCF/PCICC commands

### 5.3.4 PCICC module notebook

In this section we discuss details of the PCICC Crypto module notebook.

#### PCICC notebook GENERAL page

The panel top line indicates if you are working with a group of Crypto modules (PCICC or CCF), or with an individual Crypto module.

The panel shown in Figure 5-35 on page 192 contains a short description of the module (which can be changed), the module type, the module index and the host or group name. The PCICC can be *disabled* from this panel (this command requires a single signature) or *enabled* (dual signatures are required for enabling a module)

Although a disabled module does not provide cryptographic services to the applications, all information in the module is preserved. The intent of the disable or enable function is to permit easy control of the usage of cryptographic modules (for example, during service maintenance or non-working hours, and so forth).

**Important:** A PCICC is disabled for *all* LPARs, not just for the LPAR that the disable command was issued from.

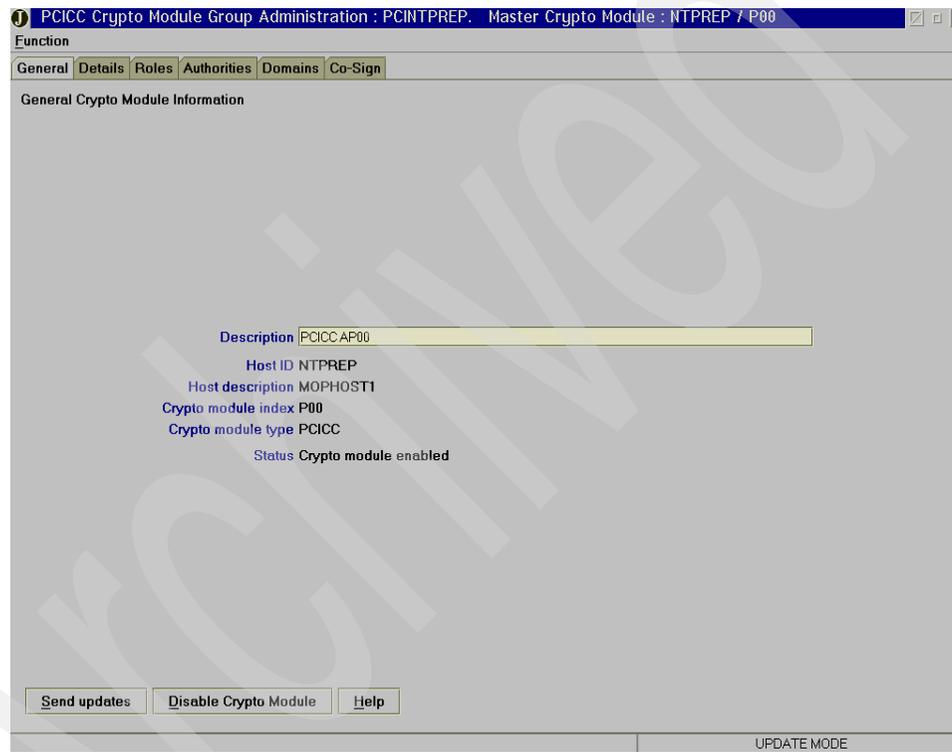


Figure 5-35 PCICC notebook general page on TKE

## PCICC notebook DETAILS page

The PCICC DETAILS page is subdivided into two sections that are individually selectable: Crypto Module and Crypto services.

The Crypto Module page contains details such as the CMID (module identifier), the module public modulus used by TKE to verify the signed replies from the Crypto module, the Transaction Sequence Number, and the hash pattern of the current DH transport key for this Crypto module.

The Crypto services page contains details on the FCV, such as the DES keys length, length of RSA keys, and base CCA availability. This information is loaded into the Crypto module at initialization of the S/390 system and cannot be changed from this panel.

## PCICC notebook ROLES page

Newly installed PCICCs (or previously zeroized PCICCs) have a predefined AUTHORITY 00 index assigned to the predefined INITADM role in the PCICC. This role can both issue and co-sign access control commands; that is, only one, single authority can perform these commands.

These commands allow you to create all the necessary roles and authorities that can be required by the customer security policy for the PCICC. Once the ultimate roles and authorities are established, remove the default values by deleting or changing authority 00 and/or by not having any authority mapped to the INITADM role. Figure 5-36 illustrates the flow of role and authorities creation in the PCICC.

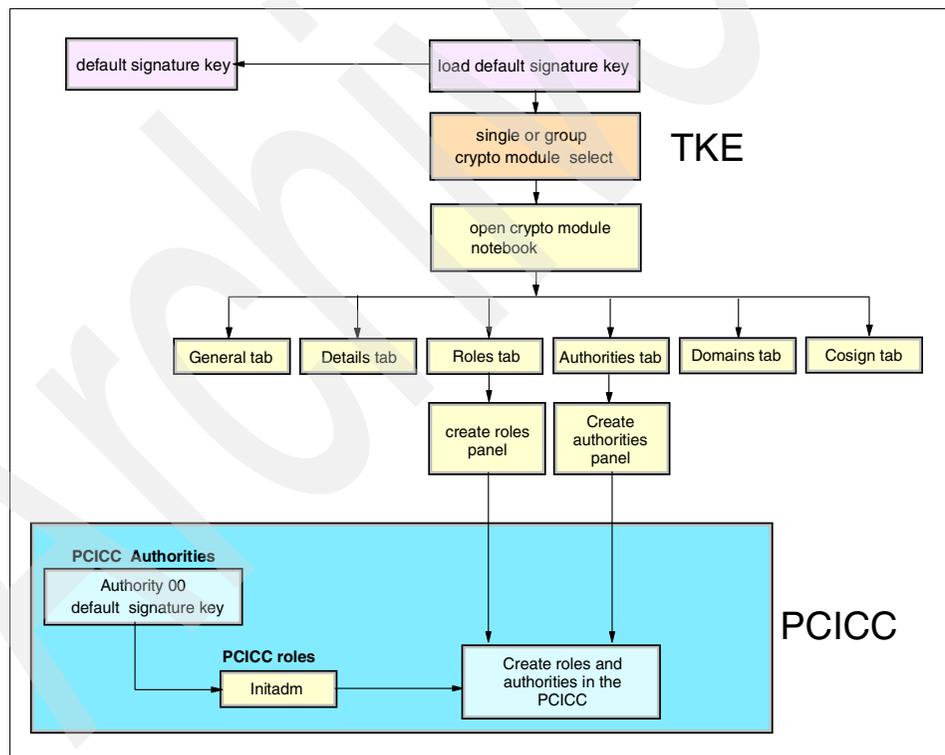


Figure 5-36 PCICC initial role and profile after new card install or card zeroize from SE

### ***Creating roles in the PCICC***

Figure 5-37 on page 195 shows the commands that can be assigned to a PCICC role for a specific domain. There are basically two types of commands: dual-signature commands, and single-signature commands.

**Note:** Roles and authorities can be defined or modified by initial default authority INITADM using default signature (to be loaded in keystore), or later on by any authority mapped to a role with Access Control command privilege (issue and/or co-sign) using its own authority key (to be loaded in keystore).

### ***Dual-signature commands***

Four commands always require *two* signatures (co-sign): one from the authority issuing the command, and one from an authority co-signing the command.

- ▶ Enable crypto card

This command is issued from the Crypto notebook general page when switching the PCICC from the disable state to the enable state.

- ▶ Access control

This command is issued with all creation, change, or deletion of roles or authorities.

- ▶ Zeroize a domain

This command is issued from the domain general page when zeroizing a domain.

- ▶ Domain controls

This command is issued from the domain control page with any update to the domain control settings.

The co-sign requirement can be alleviated by creating a role where the sign and co-sign conditions are both enabled.

**Create New Role**

Role ID

Description

**Crypto Module Enable**

Disable crypto card

Enable crypto card, issue

Enable crypto card, co-sign

**Access Control**

Access control, issue

Access control, co-sign

**New Symmetric Master Key**

Load first key part

Combine middle key parts

Combine final key part

Clear new master key register

**New Asymmetric Master Key**

Load first key part

Combine middle key parts

Combine final key part

Clear new master key register

Set asymmetric master key

**Domain Zeroize**

Zeroize domain, issue

Zeroize domain, co-sign

**Domain Controls**

Domain controls change, issue

Domain controls change, co-sign

**Domain Access**

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Figure 5-37 PCICC Roles commands definition panel

### Single-signature commands

- ▶ Disable crypto card
- ▶ Load first key part
- ▶ Combine middle key parts
- ▶ Combine final key part
- ▶ Clear new Master Key register
- ▶ Set asymmetric Master Key

Creating or changing roles can be performed only by a Crypto module authority having a profile mapped to a role authorized to perform ACCESS CONTROL commands, generally named the Crypto module(s) Administrator. The default INITADM role provides this control.

Before sending any command, the authority index signature key *must* have been loaded via the TKE main panel Functions pull-down menu or by the pop-up panel displayed when an action is selected and no signature key is loaded.

The **Default** signature key must be selected if the PCICC is new or zeroized. The signature index and authority index must be the same, so if the authority index is changed, the signature key index must be also changed, with the new index even using same default signature (use the change signature index option in the notebook function menu to make this change).

To create or change a role, the Crypto module administrator, after loading the signature key, clicks the right mouse button in the role page panel, and then selects **Create** or **Change**.

In the new role panel, the administrator defines a Role ID (if creating the role) and writes a short text description; refer to Figure 5-38.

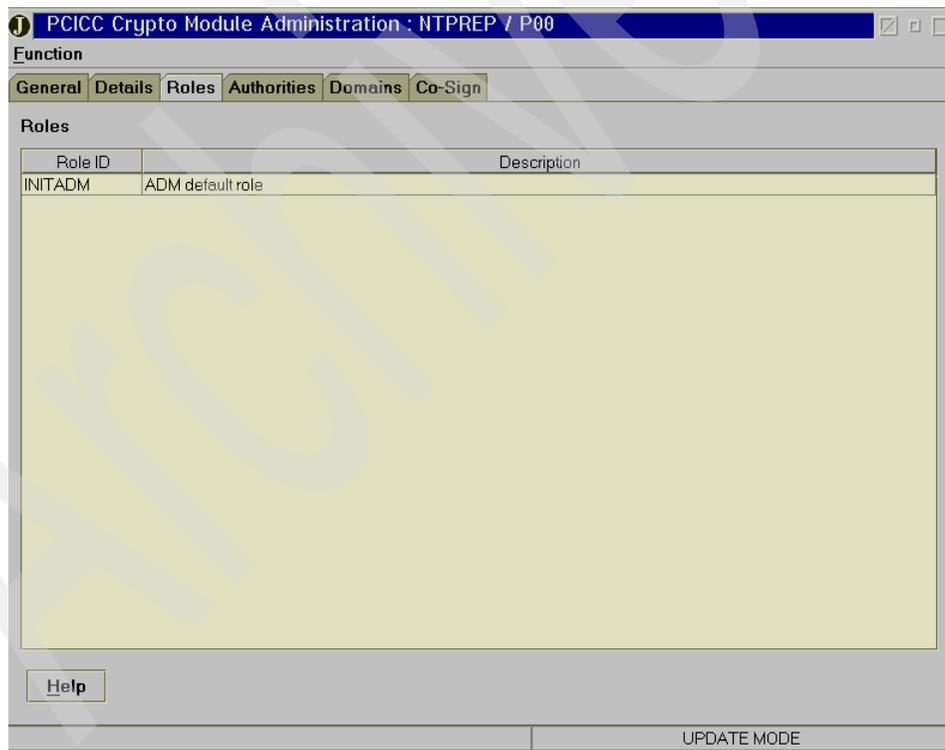


Figure 5-38 PCICC module administration notebook roles definition entry

The administrator then checks the command boxes required for the role, enters the role name with a short definition, checks target domain, and then selects **Send updates** to send the new role to the PCICC host module, as shown in Figure 5-39.

Figure 5-39 Create Role example: Domain controller role, domain 5

**Note:** Creating or changing a role is a dual-signature command, so another authority needs to co-sign, unless the administrator has been given a role where the access control commands are both issued and co-signed (that is, the initial default INITADM role has both sign and co-sign commands).

### Roles and profiles structure example

In our example, we created five roles and five authorities, using the default INITADM role. Figure 5-40 is a graphical representation of our roles and authorities structure.

- ▶ Authority 01 is mapped to the DOMAINS role. This role has the Domain Controls privileges.
- ▶ Authority 02 is mapped to the MKP1 role. This role permits you to load a symmetric and asymmetric Master Key part 1 in Domain.
- ▶ Authority 03 is mapped to the MKP2 role. This role permits you to load and combine a symmetric and asymmetric Master Key last part in Domain. This role can also set anew asymmetric key.
- ▶ Authority 04 is mapped to SYSADM1. This role has the access control privileges that permit you to create or change Roles and Authorities.
- ▶ Authority 05 is mapped to SYSADM2. This role allows you to co-sign commands previously issued by a user logged on with a profile mapped on SYSADM1.

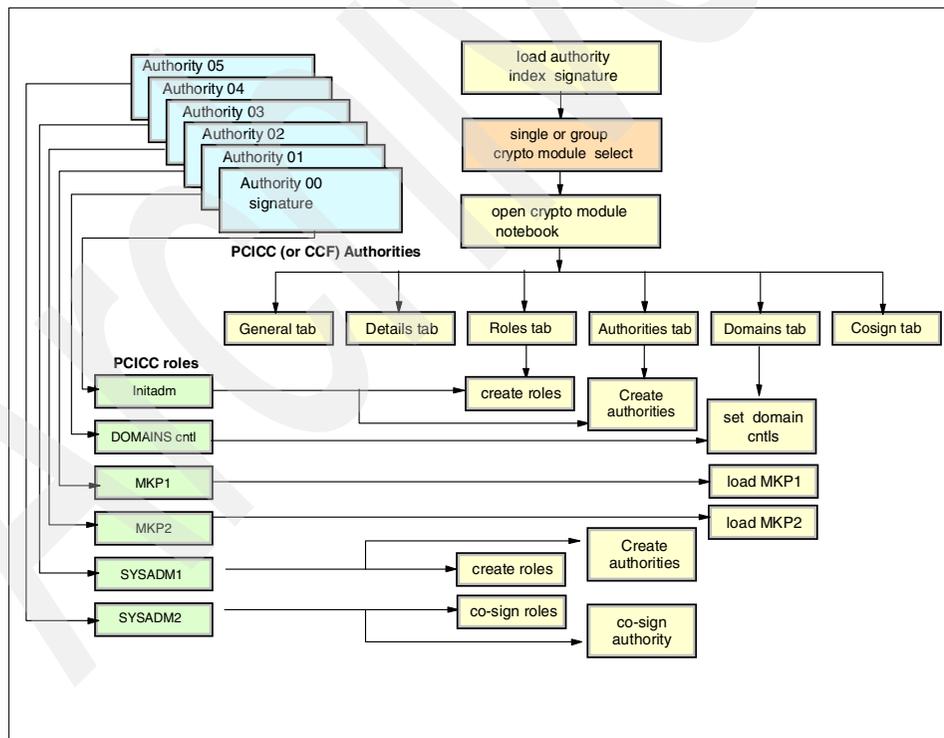


Figure 5-40 PCICC Roles and Authorities mapping example

MKP1 and MKP2 roles are shown Figure 5-41 and in Figure 5-42 on page 200.

**Create New Role**

Role ID: MKP1  
Description: Master Keys P1

**Crypto Module Enable**

- Disable crypto card
- Enable crypto card, issue
- Enable crypto card, co-sign

**Access Control**

- Access control, issue
- Access control, co-sign

**New Symmetric Master Key**

- Load first key part
- Combine middle key parts
- Combine final key part
- Clear new master key register

**New Asymmetric Master Key**

- Load first key part
- Combine middle key parts
- Combine final key part
- Clear new master key register
- Set asymmetric master key

**Domain Zeroize**

- Zeroize domain, issue
- Zeroize domain, co-sign

**Domain Controls**

- Domain controls change, issue
- Domain controls change, co-sign

**Domain Access**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>													

Buttons: Send updates, Cancel, Help

Trusted Key Entry

Figure 5-41 PCICC role MKP1: Clear new Master Key reg and enter MK part 1

**Create New Role**

Role ID:

Description:

**Crypto Module Enable**

- Disable crypto card
- Enable crypto card, issue
- Enable crypto card, co-sign

**Access Control**

- Access control, issue
- Access control, co-sign

**New Symmetric Master Key**

- Load first key part
- Combine middle key parts
- Combine final key part
- Clear new master key register

**New Asymmetric Master Key**

- Load first key part
- Combine middle key parts
- Combine final key part
- Clear new master key register
- Set asymmetric master key

**Domain Zeroize**

- Zeroize domain, issue
- Zeroize domain, co-sign

**Domain Controls**

- Domain controls change, issue
- Domain controls change, co-sign

**Domain Access**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>													

Trusted Key Entry

Figure 5-42 MKP2 role: combine key parts; set Asymmetric Master Key

SYSADMIN1 is shown in Figure 5-43 on page 201; SYSADMIN2 is shown in Figure 5-44 on page 202.



Figure 5-43 Sysadmin 1 role: access control command issue

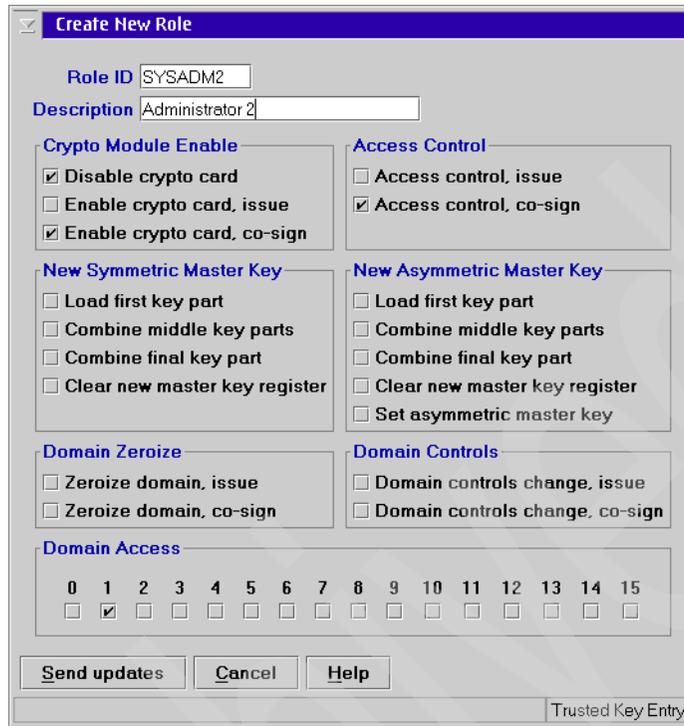


Figure 5-44 Sysadmin 2 role: access control command co-sign

Figure 5-45 shows the Roles page after the five new roles have been created.

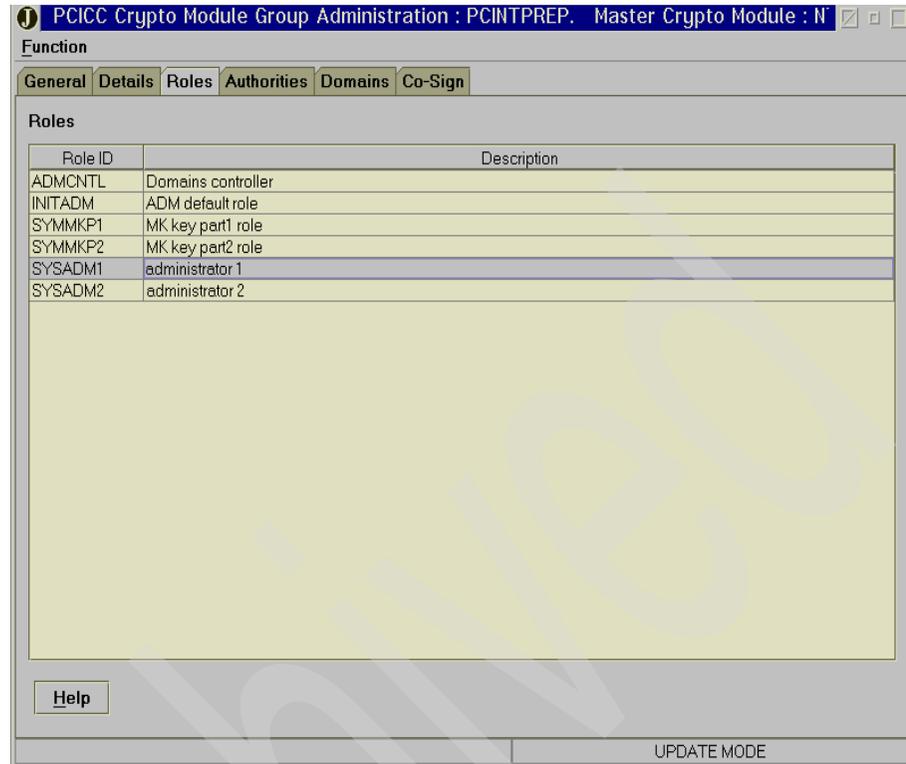


Figure 5-45 PCICC Roles page after new roles have been created

## PCICC notebook AUTHORITIES pages

The PCICC notebook AUTHORITIES pages contain the following selections:

Authorities are individual profiles mapped to one role.

### **Generate PCICC signature keys**

Before creating a PCICC new authority index, you need to generate a signature key for the index. To generate this key, select the **Authorities** tab and then click the right button of the mouse in the Authorities panel, as shown in Figure 5-46 on page 204. Then select **Generate Signature Key**.

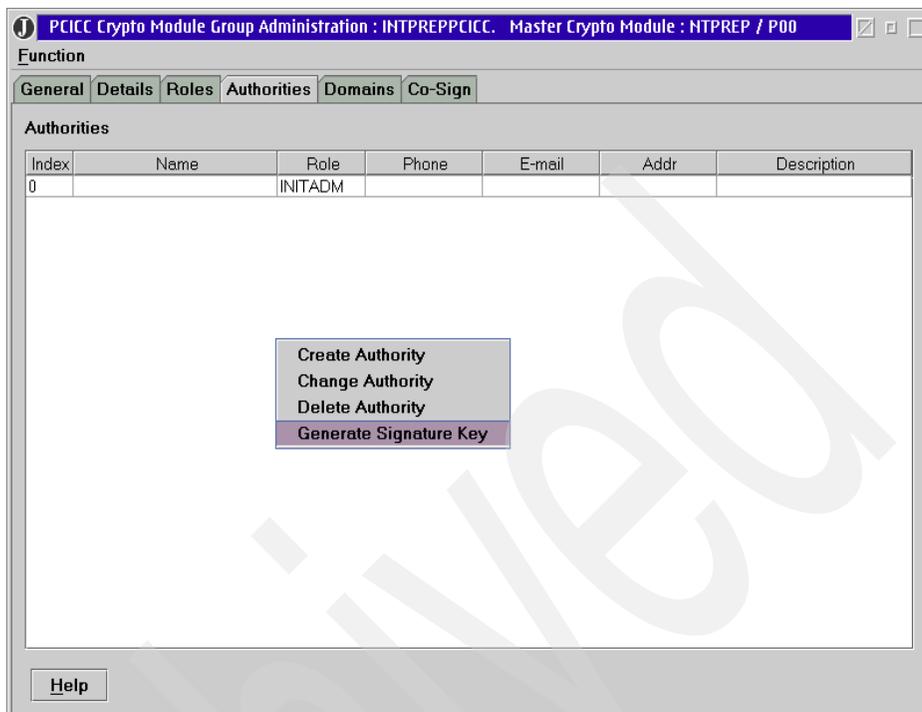


Figure 5-46 PCICC module notebook authority management main panel

On the next panel, Figure 5-47, enter the Authority index and relevant information such as name, phone number, e-mail, address and so on, and then press **Continue**.

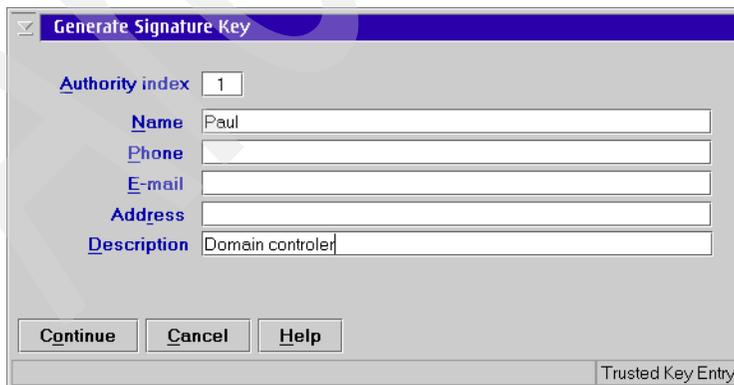


Figure 5-47 Create signature panel

Then you will be asked to save the newly generated signature on a diskette, or on a TKE hard disk, or in TKE PKA keystore. If the keys are to be saved on a diskette or hard disk file, a password is required to encrypt the key file, so keep this password in a safe place. We also recommend that you save the key again, in a different place.

**Notes:**

- ▶ Only *one* authority signature key can be saved in TKE PKA keystore.
- ▶ Be aware that an authority key loaded in the PKA keystore is loadable *without* providing a password.

You can create as many authority signature keys as needed, and you can save them for a later association with authorities. If you have decided (as we recommend) to keep the same and consistent authority indices across all coprocessors in the configuration, the same signature key files can be used when establishing authorities in the PCICC and in the CCF.

**Create a PCICC new authority**

To create a PCICC new authority, click the right mouse button in the window on the authority page, then select **Create**. A new authority signature key source is requested.

If you select:

- ▶ Keystore - the information associated with the key that you previously generated appears immediately in the window.
- ▶ Binary - you will have to specify the signature key file to be loaded from the hard disk or diskette. This authority signature key is protected by the password entered when it was generated. This is illustrated in Figure 5-48 on page 206.
- ▶ Default key - there is no associated information provided with the default key.

This panel shows the following fields:

- ▶ The authority index
- ▶ The authority information (name, phone, address, short description)
- ▶ Role - this field provides a list of previously defined roles. The authority will be mapped to the role you select.
- ▶ Authority Signature key public modulus - this part of the authority public key will be sent to the host Crypto module by pressing the Send updates button. It

will be used later on by the Crypto module to verify the signed commands sent from TKE by this authority index.

- ▶ Send updates - this is a dual-signature command and will require an authority with the proper role to co-sign.

Authority index: 3

Name: JOHN

Phone: 1200

E-mail:

Address: MOP

Description: System master key 2

Role: SYMMKP2

Signature key...  
32.63 C8D62599CF0481F6A7D566284A8FB027C2C9E7578291EF051224DF63EDB5557B  
64.95 419F6072BFFFB00B238FA281395944EE7E7E4F2904CDE92A4831447996778817  
96.127 7D0D5DF055B9C30C383F20FB750C686A345E5612D18FC712FA99693AAA8879AB  
8766D6D3F80B2864CA2FR8C536BE7294603B6CDF69EE3AA2607FR99F06E1AEE7

Buttons: Send updates, Cancel, Help

Trusted Key Entry

Figure 5-48 Create Authority: JOHN, Authority 03 mapped to SYMMKP2 role

**Note:** Authorities mapped to the role with access control privileges, such as the INITADM default PCICC role, can create, change, or delete any other role and authorities. A security measure we strongly suggest is to *replace* the INITADM role with *two* new administrator roles: the first one to issue access control commands, and the second to co-sign these commands. Or, at a minimum, change the authority 00 signature key mapped to the INITADM role and keep it in a secure place.

In our example, we created SYSADM1 and SYSADM2 roles, and Authorities 4 and 5 were mapped to these administrator roles.

### **Change a PCICC authority**

Activating this selection will allow you to make changes to the authority information and to replace the authority public modulus in the Crypto module. The authority index remains fixed.

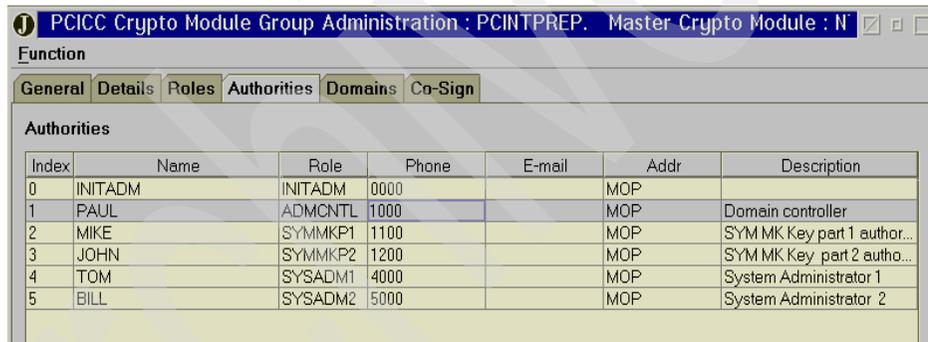
Select **Get signature key** to open a select key source window and load the signature key window. The contents of the selected key file replace the public modulus in the change authority window.

Pressing **Send updates** uploads the new information to the Crypto module.

### **Delete Authority**

PCICC authorities (unlike CCF) can be deleted from the TKE. A consistency check of roles and profiles is performed to ensure that access to the Crypto module is not lost.

Figure 5-49 shows the Authorities main panel after the five new authorities have been created.



Index	Name	Role	Phone	E-mail	Addr	Description
0	INITADM	INITADM	0000		MOP	
1	PAUL	ADMCNTL	1000		MOP	Domain controller
2	MIKE	SYMMKP1	1100		MOP	SYM MK Key part 1 author...
3	JOHN	SYMMKP2	1200		MOP	SYM MK Key part 2 autho...
4	TOM	SYSADM1	4000		MOP	System Administrator 1
5	BILL	SYSADM2	5000		MOP	System Administrator 2

Figure 5-49 PCICC Authorities panel after several authorities have been created

### **PCICC notebook DOMAINS page**

The PCICC domains entry page displays general information about each of the 16 possible LPAR domains. Selecting tabs on the right will open the specific domain general page.

Three page tabs are located at the lower left-hand side of the domain window: General, Keys, and Control. These refer to the domain general page, the domain keys page, and the domain control page; we discuss these in detail in the following sections.

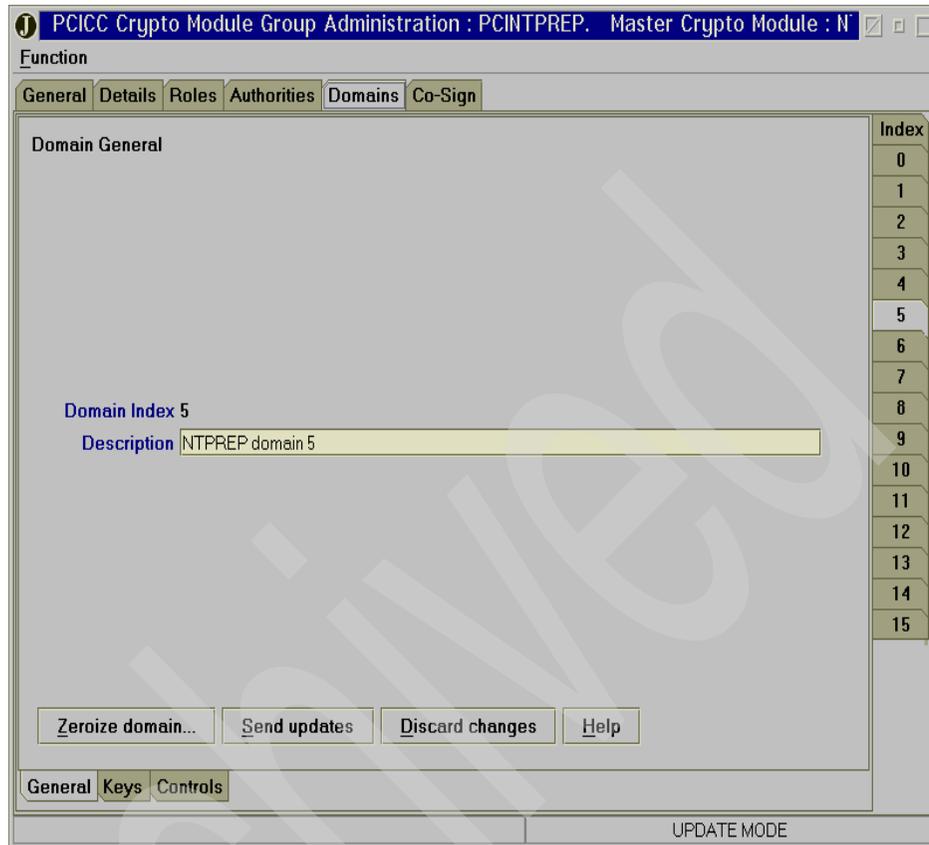


Figure 5-50 PCICC Domain 5 general page

### **Domain general page**

This page is shown in Figure 5-50. From here you can select the tasks change description and Zeroize the domain.

- ▶ To change the description, edit the description entry and press **Send updates**.
- ▶ If you choose **Zeroize a domain...**, you can do the following:
  - Erase the domain's Master Keys
  - Erase the domain's retained keys
  - Reset the domain controls

**Note:** User-defined roles and profiles are not changed.

Zeroize can be done by a user logged on with an authority mapped to a role that is authorized to perform zeroize and then co-signed.

**Important:** After domain zeroization, be aware that the ICSF panel services shown in Figure 5-52 on page 211 are in a disabled state, except for Key token change.

If these services are needed, they must be manually restored to the enabled state in the Domain Controls page.

### Domain Keys page

This page shows the Master Key status and allows you to generate, load, and clear domain key registers, as shown in Figure 5-51.

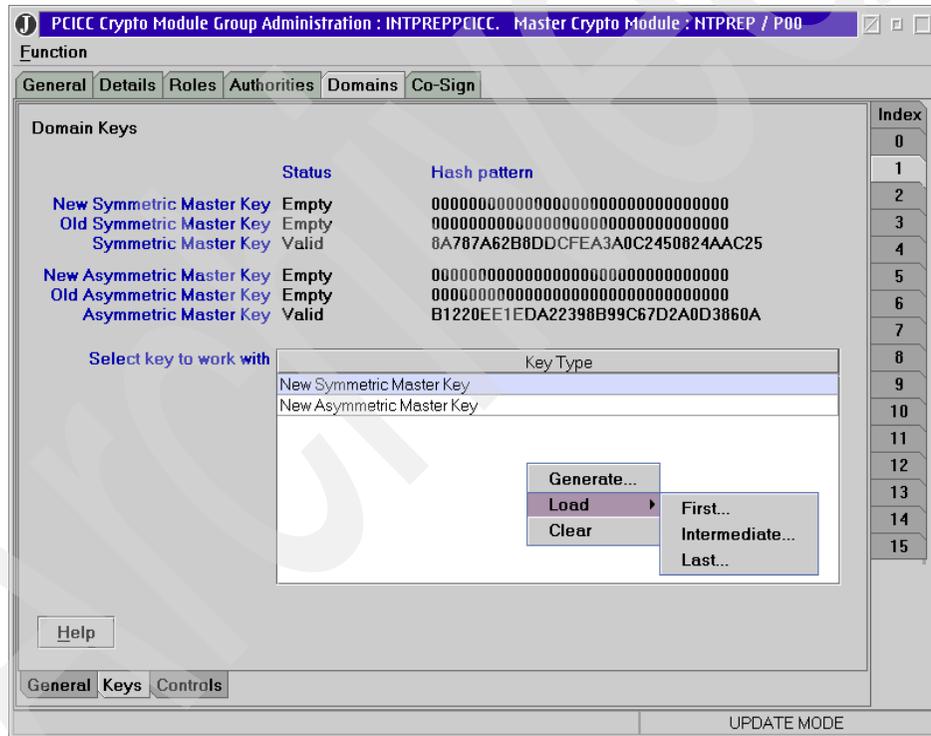


Figure 5-51 PCICC Domain Keys page

When right-clicking a key type from the key panel, the following actions become available:

- **Generate**

This will generate a key part and write it to a file. Media and file naming is a user responsibility. The number of key parts needed to create a new Master Key depends on the customer's security policy, so as many parts as needed have to be created and saved.

▶ **Clear**

This will clear the key register. Before loading first Master Key part, the key register must be cleared.

▶ **Load**

This will load a key or a key part (first, intermediate, or last) from a binary file or from the keyboard into the new Master Key register. A blind key option exists for keyboard entry as explained in 5.2.2, "TKE Workstation 4758 setup" on page 160.

▶ **Set**

Setting the ASYM-MK key will transfer the current Master Key to the old Master Key register, and the new Master Key to the current Master Key register.

***Domain control page***

This page, also named the control access point, displays the cryptographic service functions that are enabled for the domain in the ICSF ISPF panels, as follows:

- ▶ Clear New ASYM Master Key Register
- ▶ Clear New SYM Master Key Register
- ▶ Combine ASYM Master Key Parts
- ▶ Combine SYM Master Key Parts
- ▶ Load First ASYM Master Key Part
- ▶ Load First SYM Master Key Part
- ▶ Set ASYM Master Key
- ▶ Set SYM Master Key

These functions are shown in Figure 5-52 on page 211.

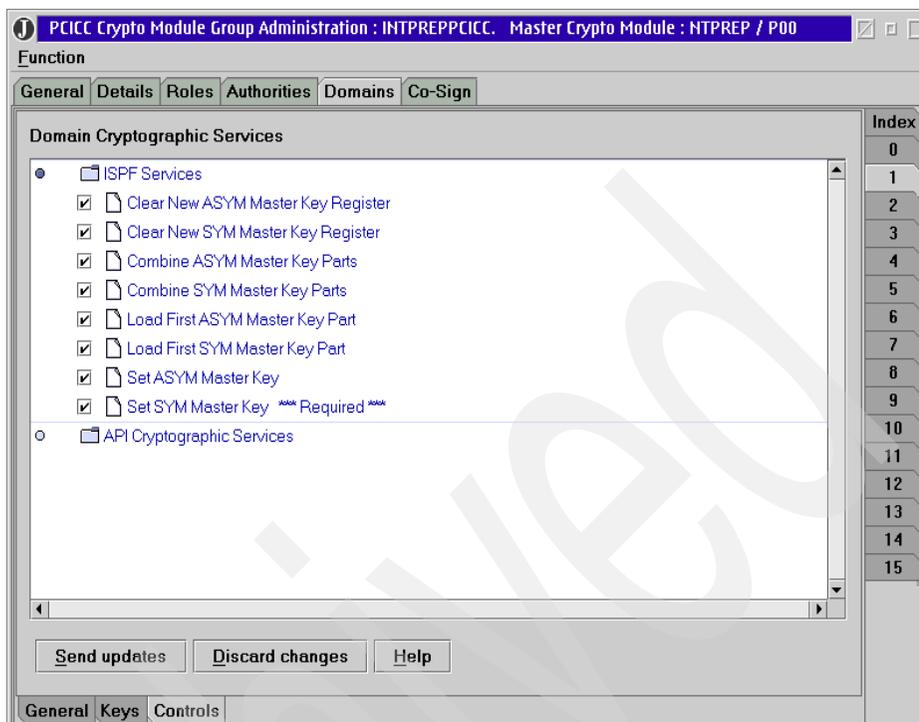


Figure 5-52 PCICC Domain 1 control panel: ISPF services

**Note:** These functions will be permitted from the ICSF panel only if the boxes are checked (but they can be performed, in any case, from the TKE). However, there is an exposure if these functions are not permitted on this panel—if the TKE fails and no other TKE is available.

Introduced with TKE 3.1, the Domain control page also displays a list of all API services available. Users must check the API boxes they want to be permitted and uncheck the API boxes they do not want to be permitted; see Figure 5-53.

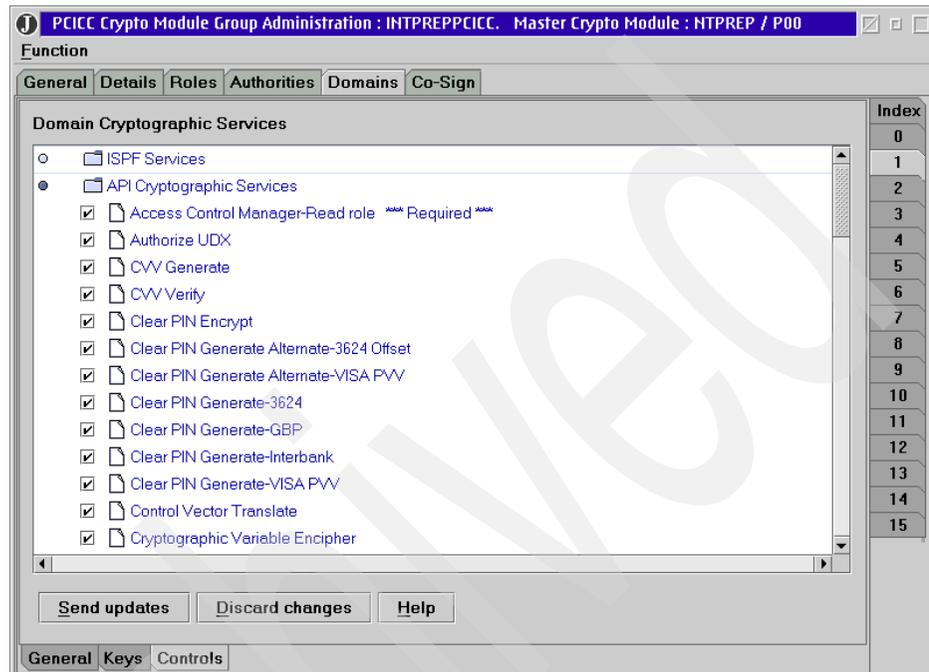


Figure 5-53 PCICC domain 1 control panel: API services

A list of available UDXs is shown in the Domain control page; see Figure 5-54.

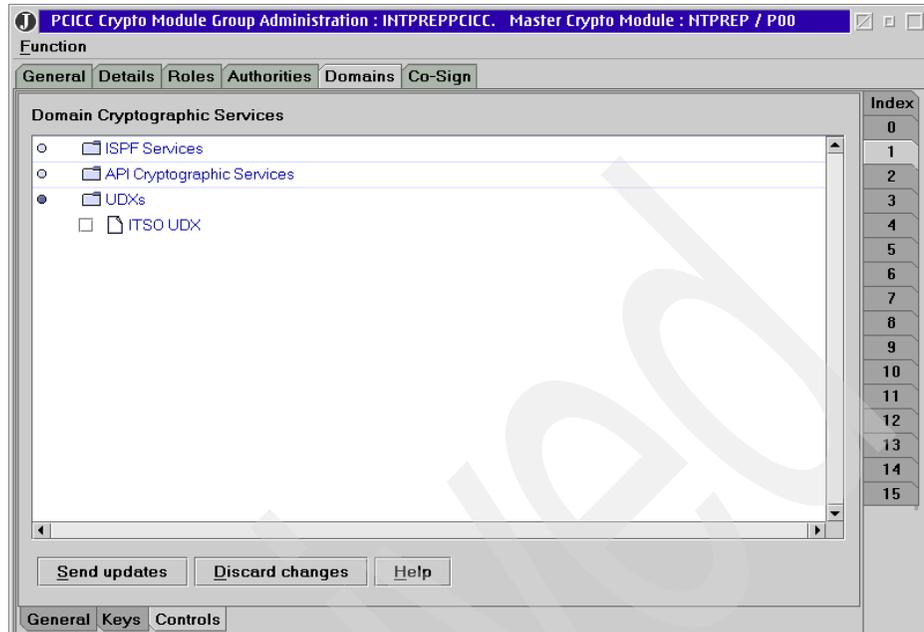


Figure 5-54 PCICC Domain control: UDX list

## PCICC Co-Sign page

For PCICC modules, some commands require two signatures before they can be executed by the Crypto module. Once an authority has issued such a command (for example, creating an authority), a pop-up message is displayed, notifying the user that the command is pending and awaiting co-sign, as shown in Figure 5-55.



Figure 5-55 Pending command message panel

This pending command can be displayed in the co-sign page. This page shows the details of the pending command and loading authority. The signature requirement field displays the authority who has already signed and the authorities that are permitted to sign this type of command; see Figure 5-56 on page 214.

- ▶ Pressing **Co-sign** initiates the signing and will require the source of the signature key to be loaded with the authority index associated with that key.
- ▶ Pressing **Delete** will delete the pending command.

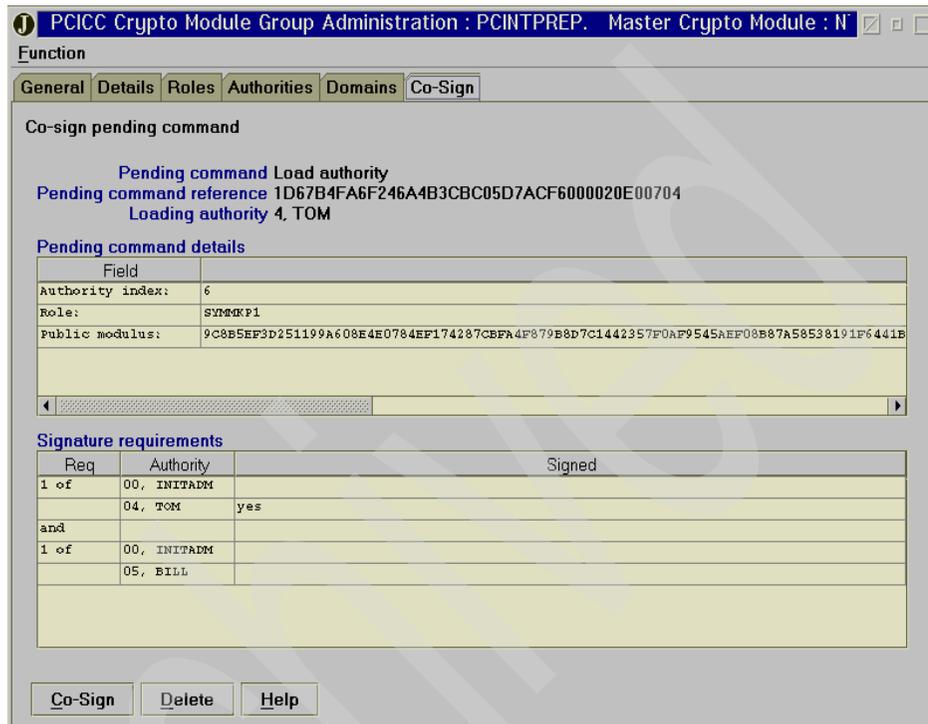


Figure 5-56 Co-sign pending for create authority command issued by TOM

### 5.3.5 Crypto CCF notebook

On the TKE main panel, you can select Crypto Module C00 or C01 or a group of CCFs to display the *CCF Crypto Module Group Administration notebook*. The notebook is the central point of control for displaying and changing all information related to a specific CCF or a group of CCF modules.

#### CCF General page

This page displays information related to a single CCF module or a CCF group (the top line in the panel indicates if you are working with a group or with a single module); see Figure 5-57 on page 215.

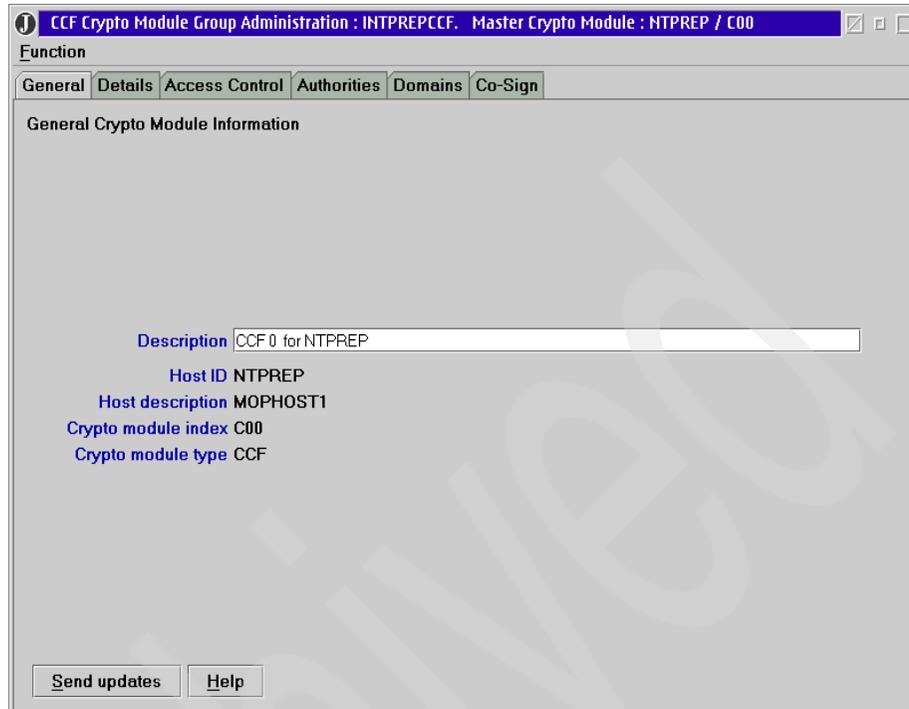


Figure 5-57 CCF General page: working with a group; C00 is master

## CCF Details page

The CCF Details page is sub-divided into five sections that are individually selectable: Crypto Module, TKE services, ICRF services, PKA services, and Key sizes.

All information in these sections is read only and cannot be changed. This page is shown in Figure 5-58.

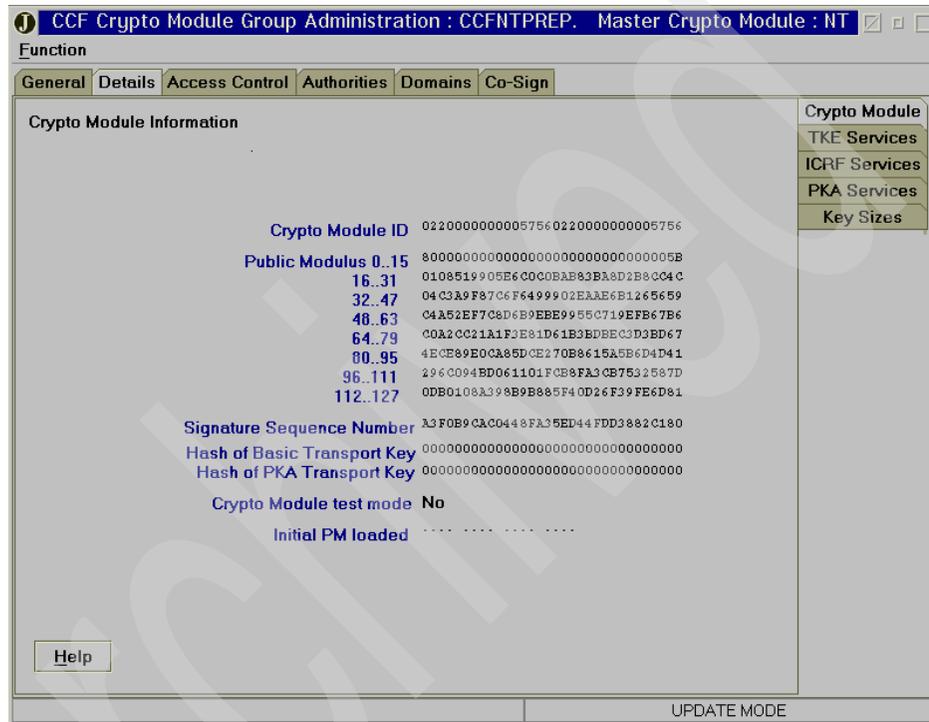


Figure 5-58 TKE CCF Crypto Module Details view

## CCF Authorities page

This page is shown in Figure 5-59 on page 217. The CCF implements 16 predefined authorities, while the PCICC allows a variable number of authorities (0 to 99). We recommend you use the same indices across the installation and use only the first sixteen indices for PCICC.

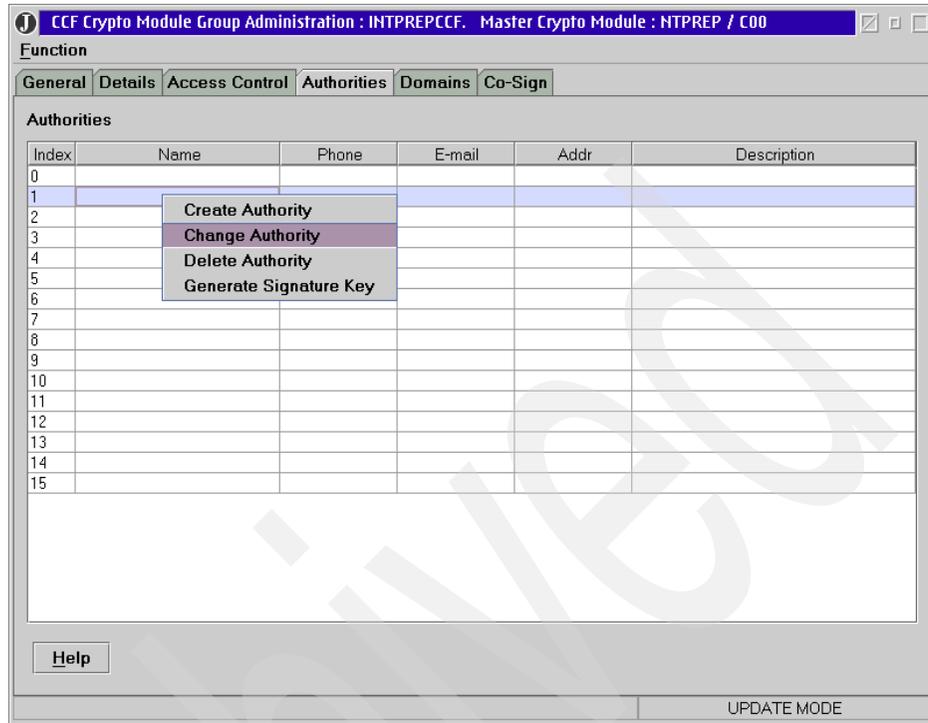


Figure 5-59 CCF authority panel: initial 16 default authorities

Unlike the PCICC, the CCF Crypto module administration does not allow you to create or delete an authority; however, change authority *is* allowed so that an authority signature key can be replaced. After CCF initialization, the 16 authorities are set up with default signature keys in the CCF. These authority signature keys should be changed to new generated signature keys, according to the site security policy.

**Important:** The default signature of authorities index 14 and 15 are not the same as the default signature of authorities index 00 to 13. Their signed commands will not be accepted by the coprocessor until new private and public keys have been established for them.

Default signature key authority index 00 will be used to change authorities and signature requirements definitions.

### **Generate signature keys**

If authorities signatures have not already been created, then you have to generate a new signature key before changing the authority definition. This

signature will be saved on either the TKE hard disk or on a diskette protected with a password. We recommend you use the same authority index and the same signature for PCICC and CCF module administration.

To create a new signature for a specific authority index, select the index, click the mouse right button and select **Generate Signature Key**. Fill in the various fields and then press **Continue**. A new panel will appear to save the new key in keystore or as a binary file protected by a password; see Figure 5-60.

**Note:** Saving the signature key in keystore is not recommended as there is no access control to the key.

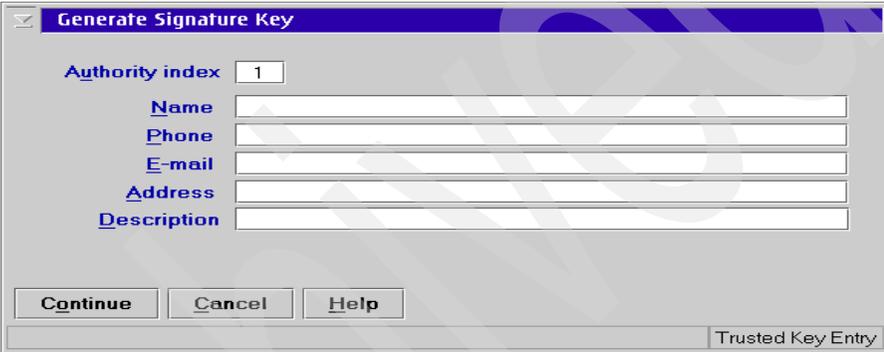


Figure 5-60 Signature key generation panel

### **Change Authority**

To customize authority definition, select an index, click the mouse right button, and select the **change** option; Figure 5-61 will be displayed.

Fill in the definition areas and select **Get Signature Key** to get a previously generated key that has been saved on diskette, hard disk or keystore. Notice that the authority index and signature index must be the same.

Then select **Send updates** to send the new authority definition to the host CCF module.



In Figure 5-62, index 00 was kept as default authority and index 01 to 05 assigned to the same authorities as in PCICC authorities definition. Signature keys are also the same. Figure 5-62.

CCF Crypto Module Group Administration : INTPRECCF. Master Crypto Module : NTPREP / C00

Function

General Details Access Control Authorities Domains Co-Sign

Authorities

Index	Name	Phone	E-mail	Addr	Description
0	default				default auth, default sign
1	PAUL			MOP	Authority for Domain Control (access ...
2	MIKE			MOP	load MK parts authority
3	JOHN			MOP	load MK parts authority
4	THOMAS			MOP	SYSADMIN1
5	BILL			MOP	SYSADMIN2
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					

Help

Figure 5-62 TKE CCF Crypto Module Authorities definition example

### CCF Access Control page

The CCF Access Control page allows you to define signature requirements, as well as the authority mask and the Domains mask; see Figure 5-63 on page 221.

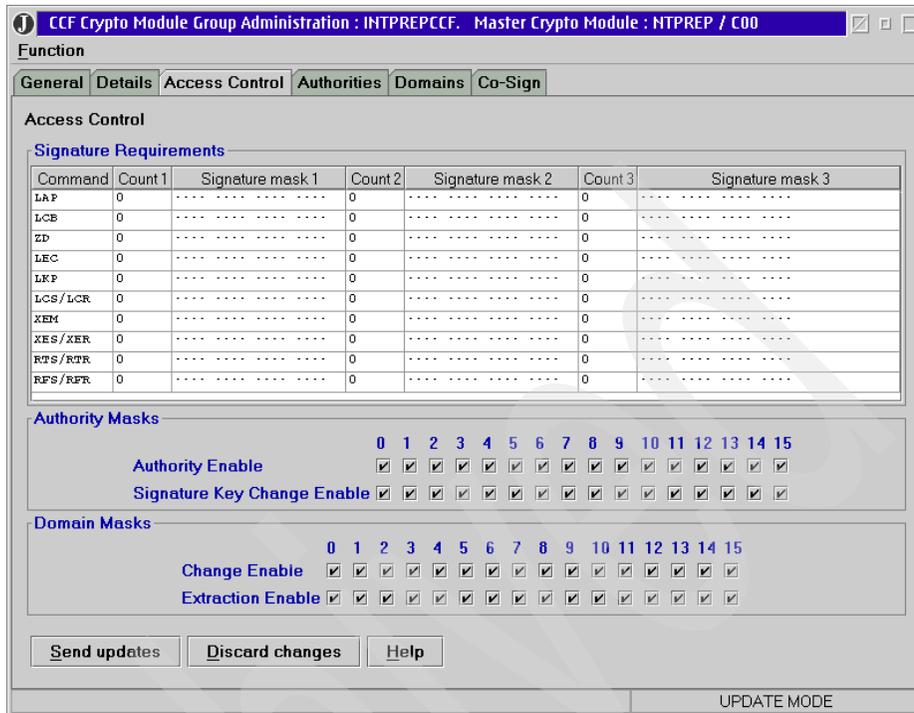


Figure 5-63 CCF Access Control initial status

### Signature requirement

The signature requirements container displays the signature requirements of each of the ten CCF multisignatures commands. There may be up to three requirements per command.

Each requirement consists of a count and a mask:

- ▶ A count (count 1, count 2, count 3) specifies the minimum number of signatures required before the requirement is satisfied.
- ▶ The mask specifies which of the sixteen authority indices can sign the command requirement (sign + means authorized).

The three requirements must be satisfied before the command is executed.

Having three requirements allows you to establish a hierarchy among authorities. Specifying a count of signature equal to zero for all three requirements means that the commands do not have any requirements.

However, to be executed, the commands must be issued by an authority that is allowed to issue signed commands (that is, enabled in the Authorization Signature Mask container). The command will not be executed if the required number of signatures is greater than the number of authorities allowed to sign.

After a new module installation or a zeroize, there is no signature requirement defined, so any command sent using the default signature will be executed immediately (refer to Figure 5-63 on page 221).

**Note:** Any multi-signature command can be initiated by any authority, being registered as a co-signer or not. These are the co-signature requirements which introduce proper control of completion of the command.

### ***The ten CCF commands***

- ▶ LAP (Load Authorization Public Modulus)- this command is issued from the Change Authority window when sending the authorization public modulus to the host Crypto module (authority signature).
- ▶ LCB (Load PKSC Control Block)- this command is issued to set the signature requirement, authority mask, and domain mask from the Access Control panel.
- ▶ ZD (Zeroize Domain)- this command is issued from the Domain General window by the Zeroize Domain function.
- ▶ LEC (Load environment control mask)- this command is issued when updating crypto capabilities from the Domain Control panel.
- ▶ LKP (Load Key parts) - this command is issued from the load and load to queue functions from the Domain Keys panel.
- ▶ LCS/LCR (Load and Combine PKA Master Keys)- this command is issued from the load and reset functions from the Domain Keys panel.
- ▶ XEM (Extract and encrypt Master Key) - this command is only for the G5 processor.
- ▶ XES/XER (Extract and Encrypt PKA Master Key) this command is only for the G5 processor.
- ▶ RTS/RTR (Reencipher to PKA Master Key) - this command is not supported by TKE.
- ▶ RFS/RFR (Reencipher from PKA Master Key) - this command is not supported by TKE.

### **Authority masks**

Authority enable - this indicates whether an authority index is authorized to issue a signed command. If the authority enabled mask is not compatible with the signature requirement, an error message will be displayed.

Signature key change enable - this indicates whether an authority index is authorized to load a new signature key in the host Crypto module.

### **Domain masks**

Change enable - this indicates whether individual domains are change-enabled (that is, able to enter keys or change cryptographic functions).

Extraction enable - this indicates whether keys can be extracted from the domain (this is no longer supported with TKE V3.1).

To send any change in the Access Control panel to the Crypto module, press **Send update**.

### **Change signature requirements**

To change signature requirements for a CCF command, double-click the mouse left button to reach the Change Signature Requirements panel; see Figure 5-64.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Signature mask	<input checked="" type="checkbox"/>															
Count1					<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>										
Count2						<input checked="" type="checkbox"/>										
Count3																

Figure 5-64 Change signature requirements panel: LAP command

Enter the number of required signatures for each line and check the boxes corresponding to the authority index allowed to sign the command, then press **OK**.

**Note:** If the number of requested signatures exceeds the number of authority boxes checked on one line, an error message will be displayed.

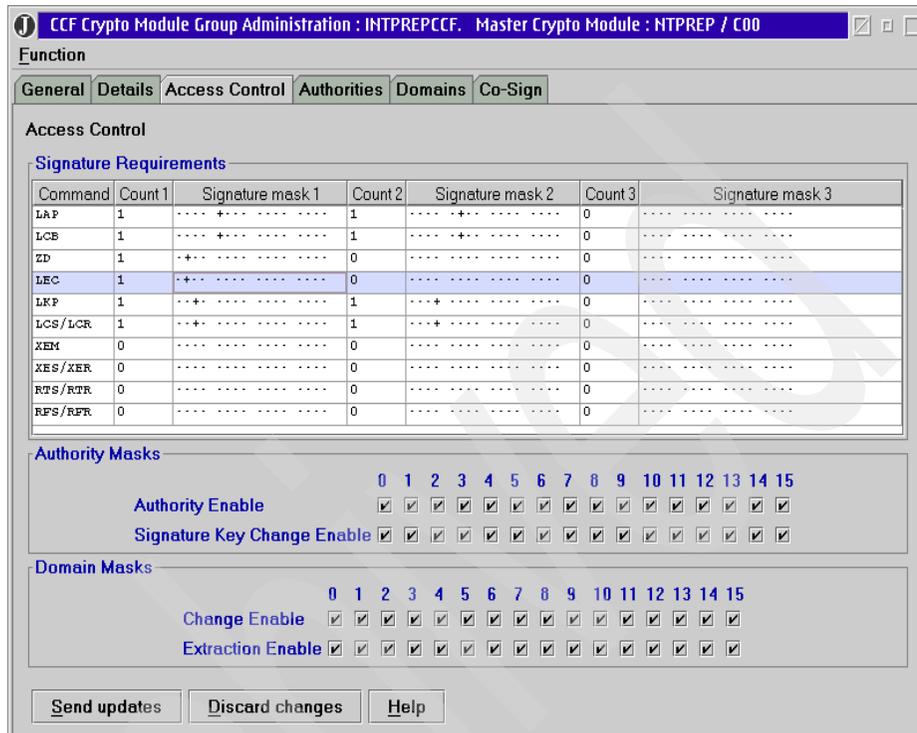


Figure 5-65 TKE CCF Crypto Module Access Control view - example

Figure 5-65 shows a simple example of signature requirements, the authority masks, and the domain masks.

It shows six commands with signature requirements among five authorities. To keep coherency between CCF and PCICC, we mapped the same authorities to equivalent PCICC and CCF function. (Refer to Figure 5-34 on page 191 for CCF/PCICC commands equivalence.)

- ▶ The LAP and LCB commands need signatures from authorities 4 and 5. Those CCF commands (load authority signature, setting of signature requirement, authority and domain mask) are equivalent to the access control command used to create roles and profiles in PCICC.

In our example, authorities 4 and 5 have equivalent functions on both PCICC and CCF modules. Authorities 4 and 5 are named system administrators.

- ▶ The LKP and LCS/LCR commands need signatures from authorities 2 and 3. Those commands (load key parts, load to queue and load and combine PKA) are equivalent to load SYM and ASYM key parts and combine functions in PCICC.

In our example, authorities 2 and 3 have equivalent functions on both PCICC and CCF modules. Authorities 2 and 3 are named Keymen.

- ▶ The ZD and LEC commands need signature from authority 1. Those commands are equivalent to Domain Zeroize and Load Domain controls functions in PCICC.

In our example, authority 1 has equivalent functions on both PCICC and CCF modules (the enable disable module does not exist for CCF). Authority 1 is named Domain controller.

Double-clicking one of the signature requirement entries brings up a dialog box allowing changes to the signature requirements for the command.

**Important:** You could be “locked out” of a CCF or a PCICC when, for example, due to an improper sequence of commands, the keys for the authorities able to sign an LCB or Access Control command do not match between the TKE and the coprocessor.

This situation can be recovered only by zeroizing the coprocessor at the Support Element (in the case of CCF, you also have to perform a system power-on Reset).

The only way to prevent this situation is to carefully track all operations involving the set up of authorities, roles, masks, and signature keys, and to understand clearly their implications in terms of synchronizing the private key and public key values between the authority and the coprocessors’ arrays.

## **CCF Domains page**

The CCF Domains page defines the domains that can have DES, PKA Master Keys, and operational keys loaded and changed from the TKE, as well as the Domain Controls set.

The page is subdivided into three parts: General, Keys, and Controls. These parts are discussed in the following sections.

### ***Domain General page***

The Domain General page appears when a domain is selected. From there, the domain can be zeroized. When a domain is zeroized from the TKE, displaying the CCF modules status in the ICSF panels shows the module in STANDBY condition; see Figure 5-66 on page 226.



Figure 5-66 CCF Domain General page

After a domain zeroize, the ICSF cryptographic functions (ECM) must be enabled in the Domain Control panel to allow loading of the Master Keys through the TKE Workstation.

Providing access to the clear Master Key entry function via the ISPF panels should be based on the site security policy.

**Note:** The “Zeroize domain” function in the Domain page is different from the zeroize in the Crypto management panel on the service element panel in that only configuration data and keys of the selected domain will be cleared.

After zeroizing, remember to restore the settings in the Domain control panel to conform to the site security policy.

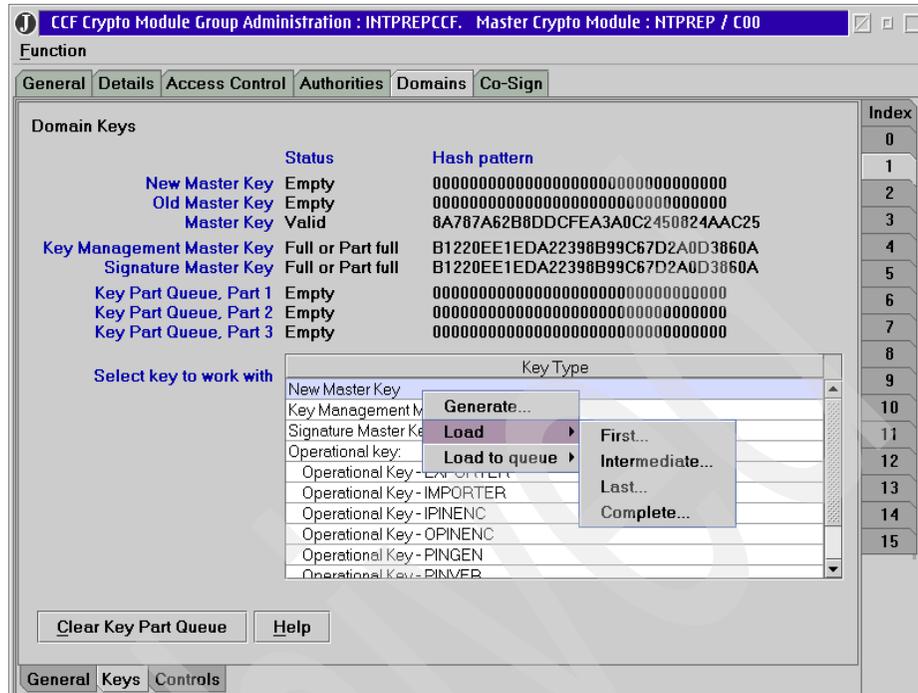


Figure 5-67 CCF Domain keys page

### Domain keys page

As shown in Figure 5-67, selecting the **Keys** tab displays Master Key and key parts queue status information, and allows you to generate, load, and clear host domain key registers. It also allows you to create and load operational keys and RSA keys.

#### Key types:

- ▶ NEW MASTER KEY
- ▶ KEY MANAGEMENT MASTER KEY - a DES key used to protect PKA keys.
- ▶ SIGNATURE MASTER KEY - a DES key used to protect signature keys. To maintain consistency with PCICC, we recommend that you use the same key management Master Key and Signature Master Key.
- ▶ OPERATIONAL KEYS:
  - EXPORTER key - exporter key encrypting key.
  - IMPORTER key - importer key encrypting key.
  - IPINENC key - Input PIN encrypting key.
  - OPINENC key - Output PIN encrypting key.

- PINGEN key - PIN generation key.
  - PINVER key - PIN verification key.
  - IMP-PKA key - a DES key used to encrypt, then transfer RSA keys from TKE to host Crypto module. ICSF cannot use this key for any purpose other than importing an RSA key from TKE.
- ▶ RSA key

### **Key functions**

You select key type by using the mouse left button, and then clicking the right button to get the available functions.

- ▶ GENERATE - New Master Key parts, key management Master Key parts, signature Master Key parts, operational or RSA keys are generated and saved on a selected media at the TKE. The naming convention of saved files naming convention is a user responsibility.
- ▶ LOAD or LOAD TO QUEUE - A key part is read from a selected source at the TKE and uploaded in the Crypto module. Depending on the key type, the user can select whether the load is done directly to the crypto chip register (LOAD) or to the key part queue (LOAD to QUEUE).

Use the load **First** option to load first key part, use the load **intermediate** option to load second key part (if more than two key parts are needed), and finally use the load **Last** option to load the last key part. The option Complete can be used if only one key part is needed.

- ▶ CLEAR - This will clear key management or signature Master Key.
- ▶ LOAD TO KEYSTORE - Load Exporter, Importer or IMP-PKA key in TKE key storage (this key will be used to encrypt or decrypt keys to be transferred between TKE and the host Crypto module).
- ▶ ENCIPHER - This function will encrypt an RSA key created on TKE using an existing IMP-PKA key, before transferring this RSA key to the host.
- ▶ LOAD to PKDS - This function will send an enciphered RSA key to the host PKDS.
- ▶ LOAD to DATA SET- This function will send an enciphered RSA key to the host data set.

### **Generation and load to queue of New Master Key**

Master Key is a DES key used by the host cryptographic module to protect keys stored in CKDS.

ICSF and the crypto chip have been designed to handle Master Keys that are comprised of two or more key parts. Users decide how many key parts in total they want the Master Key to be made of (this depends on the customer's security policy).

Select **generate** for the New Master Key category “save generated keys parts into a output media”. Repeat generate and save as many times as the number of key parts needed to create the NMK according the site security policy.

Select **load to queue** and retrieve the saved key parts from media for loading the key parts into the queue part register of the crypto chip. Load the **First**, **intermediate** (if needed), and then the **last** key part. The load **Complete** option can be used if only one key part is needed to create the new key.

Next, perform these tasks from the ICSF ISPF panels:

- ▶ Load the key parts from the load queue into the DES Master Key auxiliary register, which is then the DES New Master Key Reg.
- ▶ Reencipher CKDS to new Master Key (if changing Master Key).
- ▶ Set the DES Master Key register from the New Master Key reg.

**Note:** The auxiliary master key register can contain either the New Master Key or an old Master Key. Therefore, a New Master Key and an old Master Key cannot coexist at the same time in the Crypto module.

If an old Master Key exists in the auxiliary register, it will be lost when you enter a new one or if you reset the New Master Key register.

### ***Generation and load of Key Management or Signature Master Keys***

These Master Keys are DES keys used to protect PKA management or Signature keys stored in PKDS. To maintain consistency with PCICC modules, we recommend that you use the same key for both PKA management and Signature keys (KMMK = SMK).

The generation process is the same as for New Master Key parts; the only difference is that key parts (first, intermediate, last) are directly loaded in the register.

**Note:** PKA callable services must be stopped before loading new KMMK or SMK. Keys stored in PKDS and protected by the old KMMK or SMK can be reenciphered with a new KMMK or SMK by ICSF beginning with z/OS V1R2—but only if a PCICC card available for the domain.

### ***Generation and load to queue of operational keys***

Operational keys are DES keys intended for use by user applications, and they are to be kept encrypted under the Master Key in the CKDS.

Like Master Keys, these keys are also made of several parts (the number to be determined by the customer) generated and saved from TKE on output media, and then loaded from the TKE to the crypto chip load queue.

Exporter keys and IMP-PKA keys can also be loaded in the TKE keystore to be used as operational keys by TKE.

### ***RSA key generation***

The RSA key generation process allows you to generate an RSA key pair at the workstation, and then send this key pair to any PKDS. If the PKDS is designated to receive, the RSA key is attached to the TKE ICSF host z/OS image, and then directly loaded in the PKDS.

If the designated PKDS is attached to another z/OS image not running the TKE Host ICSF instance, the RSA key must first be loaded into an z/OS PDS member from the TKE and then migrated to the destination PKDS using the ICSF ISPF panels.

The RSA key is transferred from the TKE to the ICSF encrypted under a special DES encrypting key called IMP-PKA key (which is used as an exporter key by TKE).

The IMP-PKA created at the TKE is transferred to the host ICSF CKDS, and then used as a transport key (importer) by the host crypto to decrypt the received RSA key.

The sequence to generate and transfer an RSA key is as follows:

- ▶ Create and install the IMP-PKA key on TKE and host side.
- ▶ Create the RSA key.
- ▶ Transfer the RSA key, protected by the IMP-PKA previously created.

### ***Generation and installation of the IMP-PKA key***

An IMP-PKA key is a DES key, part of the operational keys, and it can be comprised of several key parts. Therefore, as many parts as needed must be generated and saved on output media.

The IMP-PA Key management (key parts generation and loading) process is started by selecting functions for IMP-PKA in the operational keys group; refer to Figure 5-68 on page 231.



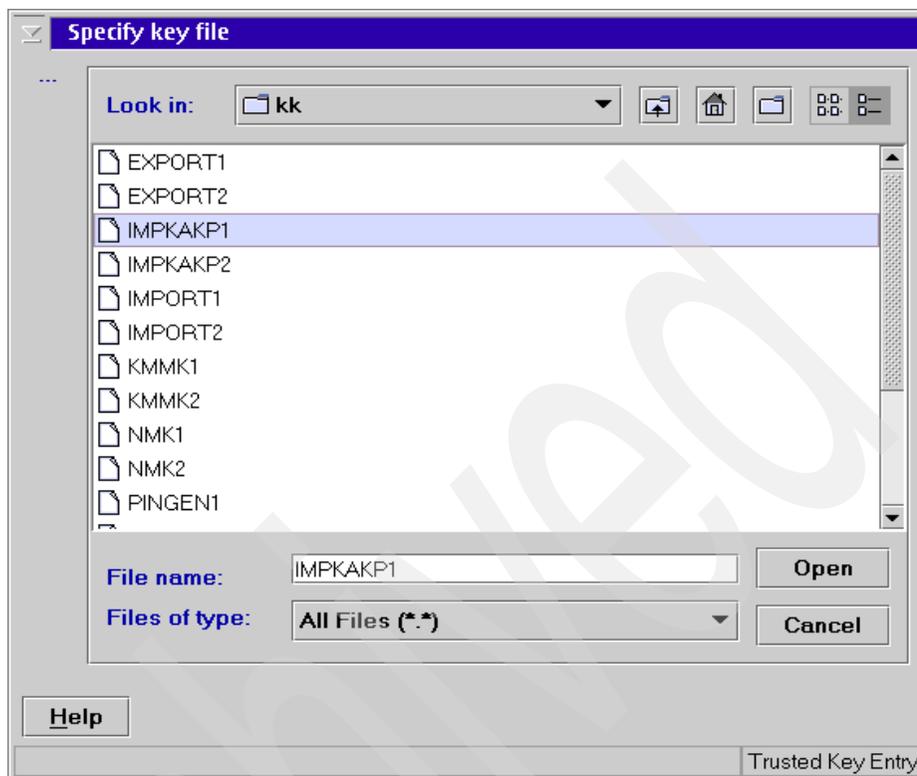


Figure 5-69 IMP-PKA parts: IMPPKAP1 and IMPPKAP2

The IMP-PKA key parts are then loaded in the host Crypto module using **load to queue**, as with any other operational key. From the ICSF ISPF panels, the IMP-PKA key will then be installed in a user-selected CKDS under a user-selected label.

The key labelled IMPPKA1 was created and loaded in the host, using IMPPKAP1 and IMPPKAP2 as the first and last key parts. This was accomplished by the following process.

Select the IMP-PKA key line entry, click the mouse right button, select **load to queue**, then the first part, and click the mouse left button. Select the key part from media, and after verifying, press **Load**. This will send the key to the host cryptographic processor. Then go to the ICSF ISPF panel to load this first part and create a key with the desired label

Repeat this step for the last key part (or intermediate, if needed), then save the new, labelled IMP-PKA in CKDS.

The IMP-PKA, in its exporter form, must also be installed at the TKE in preparation for the transport of the RSA key. It will be installed in the TKE-owned file called the DES key storage (the IMP-PKA will be used as an exporter key by the 4758 card in the TKE), using the same label as was used on the host side.

In our example, the key labeled IMPPKA1 was created using IMPPKAP1 and IMPPKAP2 part keys. This was accomplished by the following process.

Select the IMP-PKA key line entry, click the mouse right button, select **load in keystore**, then select the first part and click the mouse left button.

Figure 5-70 on page 234 shows the load in the TKE keystore main panel. On this panel, we enter the new IMP-PKA key label in the workstation key label area.

Selecting **For RSA Key generation** will make this TKE IM-PKA exporter key not exportable; selecting option **For RSA key enciphering** will allow this key to be exported. We press **Continue**, and the next panel requested that we select the key part from input media (refer to Figure 5-69 on page 232). On the next panel, after verifying, we press **Load key**.

When the first part completes, we load the last key part (or intermediate, if needed).

At this point, the IMP-PKA key labelled IMPPKA1 is available in the Host CKDS as an importer key and is available on the TKE side as an exporter.

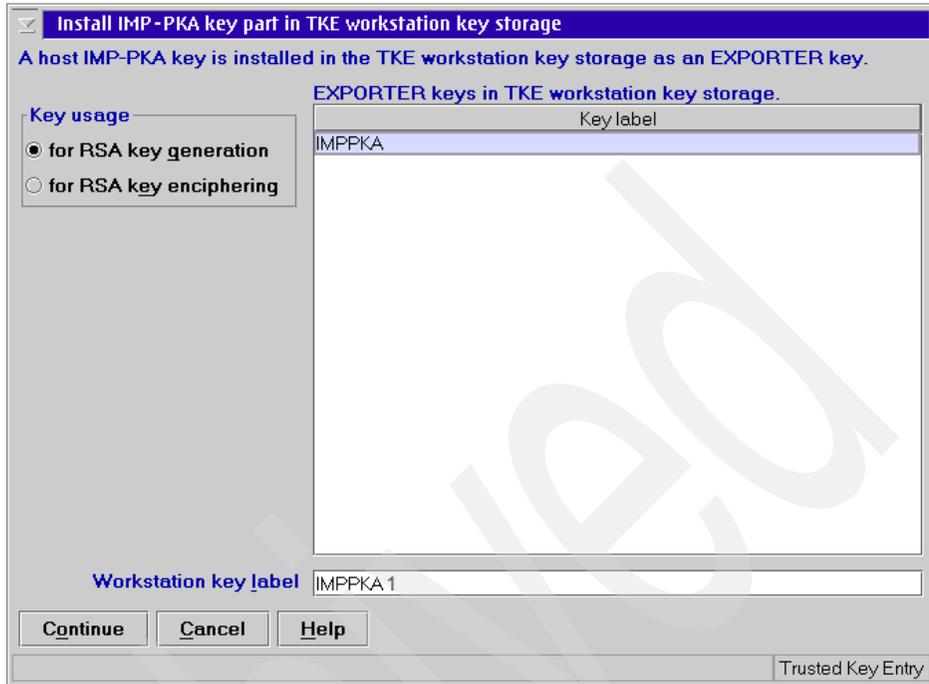


Figure 5-70 IMP-PKA load to TKE keystore: new IMPPKA1 to be loaded in keystore

### **Generate RSA key**

The RSA key will be generated and enciphered using the previously created IMP-PKA key; see Figure 5-71 on page 235.

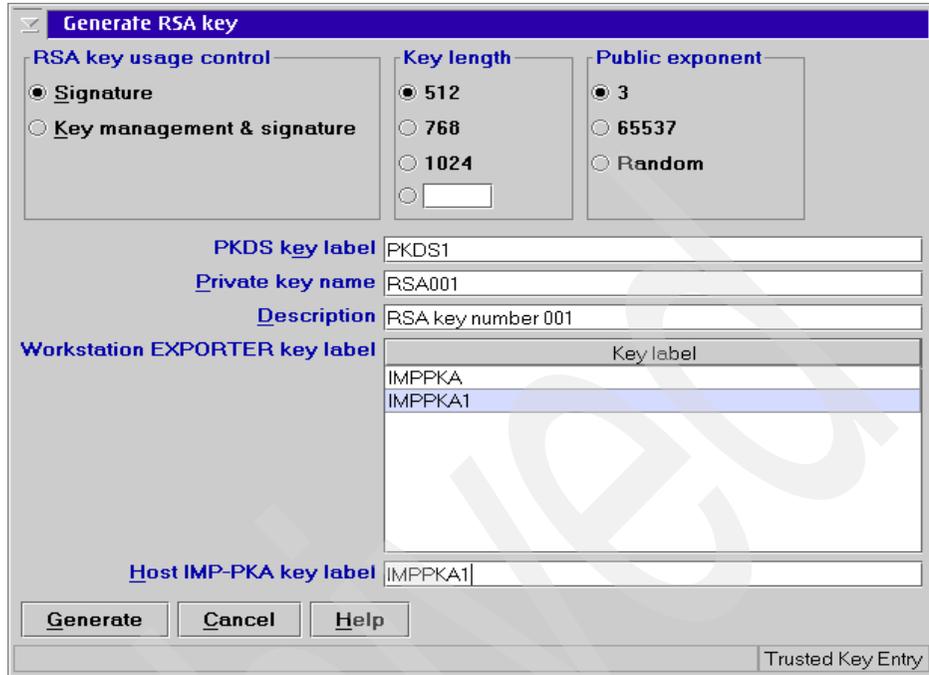


Figure 5-71 RSA001 will be created and protected by IMP-PKA key labeled IMPPKA1

To call this function, users must select the RSA key line, click the mouse right button, and then select **Generate**. On this panel, users must enter the RSA key type (SMK or KMMK), key length and exponent, the PKDS host name, and the RSA key name with a short description.

Users will also define here the TKE exporter key label (the IMP-PKA key loaded in the TKE keystore) and the host IMP-PKA label to be used to protect the RSA key.

Press **Generate** to create the RSA key. This key must then be saved on the TKE output media. The file naming convention is a user responsibility.

In our example, RSA key RSA001 has created, and protected by the already implemented IMPPKA1 IMP-PKA key for target PKDS1.

Load RSA key to PKDS

PKDS key label PKDS1

Private key name RSA001

Description RSA key 0001

Workstation EXPORTER key label IMPPKA1

Host IMP-PKA key label IMPPKA1

Send Cancel Help

Trusted Key Entry

Figure 5-72 Transfer RSA key information pane

### **Transfer RSA key**

The RSA key is sent to the host by using the options Load to Data set or Load to PKDS. From the RSA key function pull-down menu, select send to Data set or to PKDS. After you provide input media, Figure 5-72 will appear. If the data is correct, press **Send**.

**Note:** Use of the Load to PKDS option is restricted to loading RSA keys to the active PKDS at the z/OS image where the TKE host transaction program resides.

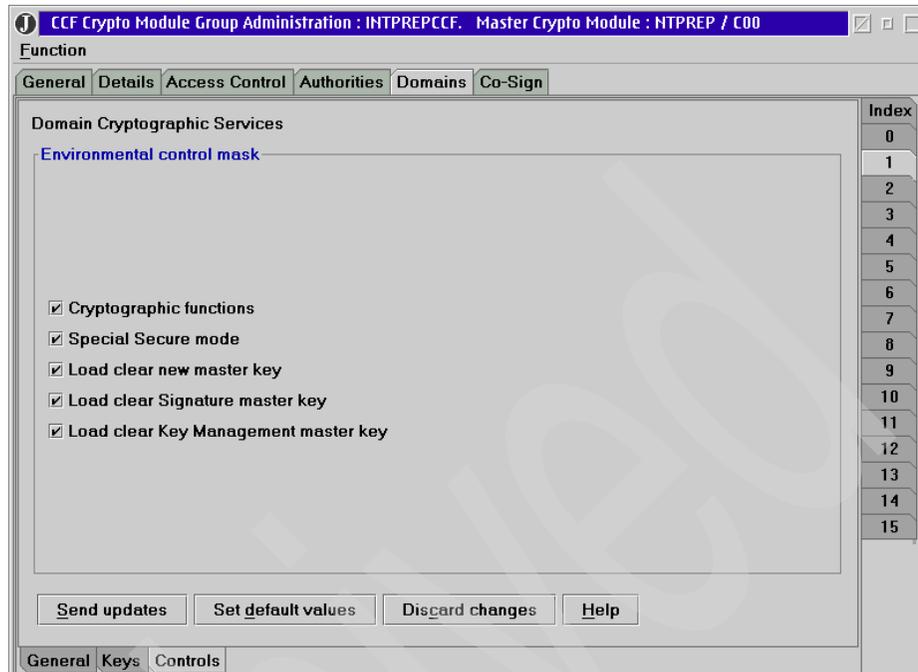


Figure 5-73 CCF Domain control page

### **Domain control page**

**Controls** is the Crypto function enablement panel. It permits enabling/disabling Cryptographic functions, Special Secure Mode, and access to the clear New Master Key, SMK and KMMK entry panels in the environmental control mask (ECM).

As seen in Figure 5-73, Cryptographic functions *must* be enabled in order to access the Crypto coprocessors.

Enabling SSM, or Special Secure Mode, allows you to input clear keys and generate clear PINs. The key generator utility program (KGUP) utility uses the SSM to generate and input clear keys.

If clear entry functions are disabled on this panel, entering clear keys from ICSF panels will not be allowed; this can only be done from the TKE.

Depressing **Set default values** will enable only the cryptographic functions; all other boxes will be unchecked.

Pressing **Send updates** will send changes to the host cryptographic module.

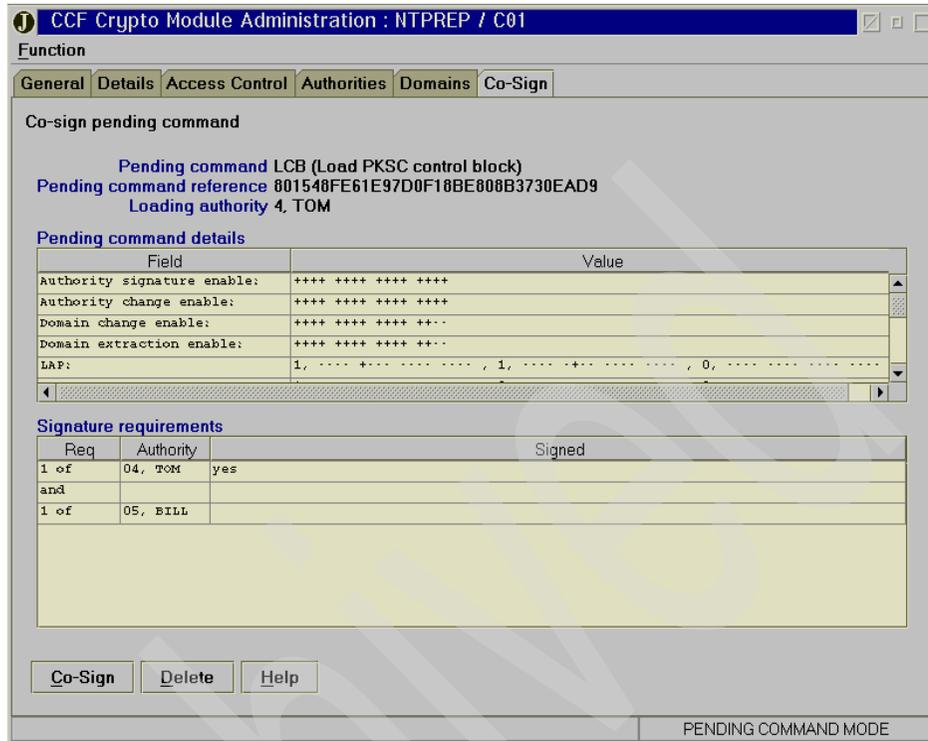


Figure 5-74 TKE CCF Crypto module co-sign view

## CCF co-sign

In our example, shown in Figure 5-74, signatures from authorities 04 and 05 are required for an LCB command. Authority 04 has issued the LCB command and the command is pending, waiting for the authority 05 signature.

To load the signature of one of the authorities allowed to sign the command, depress the sign command button, select the signature source, and change the index signature.

When the number of required signatures is fulfilled, the command is executed.

A pending command can also be deleted.

### 5.3.6 Backing up the TKE files

There are a number of files that should be backed up when the TKE initialization tasks have been completed. These include any TKE roles and profiles defined

using the CCA support program CSUECNM tool and saved to disk. It also includes the authority signature keys saved to binary files.

## Workstation files

You should systematically back up files on your workstation. Double-click the **Backup** icon and follow the instructions given there. The TKE backup diskette is labelled ACTKEBKP.

- ▶ HOST.DAT contains the definitions for the host sessions and related data. It also contains the CMID and public modulus key for each module.
- ▶ GROUP.DAT contains definitions for host modules groups.
- ▶ TKE.INI contains customization information for TKE.

These files should be backed up whenever definitions for any of the above are changed.

- ▶ DESSTORE.DAT and DESSTORE.NDX.

These are the DES key storage areas used to hold IMP-PKA keys for encrypting RSA keys and EXPORTER keys used by the 4753 migration facility.

- ▶ PKASTORE.DAT and PKASTORE.NDX.

These are the PKA key storage areas used to hold one authority signature key.

Additionally, if you have saved authorities signature keys or Master Key parts to a binary file on the hard drive, they should be backed up as well. If you have previously generated and loaded DES Master Key parts during the TKE IBM 4758 initialization and saved them to the hard drive, they must be backed up. Since these files are named by you, there is no automated procedure to back them up. You must manually copy the files to the TKE binary diskette that is shipped with the TKE Workstation.

**Note:** Do *not* use the backup diskette for this purpose.

## Host files

The Crypto module (CM) data set on the MVS Host system must be backed up. The name of the data set is defined in the parameters of the Host Transaction Program.

This data set contains definitions for the Crypto modules, domains, and authorities. It is updated any time the user makes changes in the TKE application windows and Crypto module notebooks.

## 5.4 4753 Key Token Migration facility

A facility is provided in the TKE V3.1 to migrate 4753 key tokens into the ICSF cryptographic key data set. It runs a migration utility in the TKE Workstation and provides the transfer facility, so that the migrated tokens can be transferred into the host CKDS.

We did not practice with this utility during our project; therefore, this information is provided for reference only. For further information, refer to Appendix F of the *TKE Workstation User's Guide 2000*, SA22-7524, which provides the detailed procedure.

## Support functions

In this chapter we describe the additional support functions provided by z/OS to ICSF users as of z/OS V1R3.

## 6.1 RACF access control to ICSF services

RACF profiles in the CSFKEYS class can be used to protect access to the keys stored in the CKDS or PKDS. Access control is performed, by ICSF, on the basis of the key label and the RACF userid, or group name, of the application calling ICSF. These profiles are user generated and do not relate to any specific level of ICSF.

Another class of RACF profiles, CSFSERV, is used to control access to ICSF services, including services provided through ISPF panels as well. Access control is again performed by ICSF on the basis of the service name, as described in the relevant chapter in the *ICSF Administrator's Guide* and in the calling application RACF userid, or TSO/E userid for the ISPF panels, or group name.

The restrictions are enforced only when the CSFSERV general resource class is active; to activate the class, enter the following commands:

```
SETROPTS CLASSACT(CSFSERV)
```

```
SETROPTS RACLIST(CSFSERV)
```

After permitting specific users to the profile, the RACF administrator must refresh the in-storage profiles by issuing this command:

```
SETROPTS RACLIST(CSFSERV) REFRESH
```

### 6.1.1 New profiles in the CSFSERV class

For information, the following profile has been added in the CSFSERV general resource class at z/OS V1R2:

**CSFPKDR** This protects the PKDS reencipher and activate functions when called either via the ISPF panels or the CSFPUTIL utility program.

**Note:** There are no new CSFSERV profiles specific to z/OS 1.3, and the reader is encouraged to consult the *ICSF Administrators Guide* for further information on existing profiles.

**Important:** The new AES encrypt and decrypt services (CSNBSYE and CSNBSYD) are not protected by CSFSERV profiles.

## 6.2 Crypto usage measurement

Crypto usage was so far reported in the following SMF records

- ▶ type 30 (Common Address Space Work), with field SMF30CSC in the Processor Accounting Section reporting the number of cryptographic instructions executed on behalf of the caller.

**Note:** Each time ICSF issues a hardware instruction, this count is incremented. This means that in some cases, like bulk encryption, the count would be incremented by more than one, and for other operations, like a PIN verification, the count would increment by one.

- ▶ type 82 (ICSF Record), reporting information about the events and operations of ICSF. Besides subtype 17 (which provides some information on the PCICC utilization), all other subtypes report on administrative kinds of operations

Note that beginning with z/OS V1R3, sample jobs are available in SYS1.SAMPLIB to format type 82 records: CSFSMFJ and CSFSMFR. See Figure 6-1 on page 248 for an example of this formatting.

- ▶ type 70 (RMF Processor Activity). At z/OS V1R2 with APAR OW49808, record type 70 has a new subtype 2 for measurements of cryptographic processors. The data of the traditional record type 70 becomes part of record type 70 subtype 1.
- ▶ type 72 (RMF Workload Activity and Storage Data) reports on class period and crypto support by WLM, beginning with z/OS V1R2 with APAR OW49808.

### 6.2.1 SMF record type 82

#### ICSF Initialization (subtype 1)

When ICSF starts, ICSF writes to subtype 1 after initialization is completed. Subtype 1 describes the values of installation options that are specified in the installation options data set. Subtype 1 contains the following information:

- ▶ Special secure mode (SSM) option
- ▶ Key authentication (KEYAUTH) option
- ▶ Security Server (RACF) checking of Supervisor State and System Key callers (CHECKAUTH) option

- ▶ Compatibility mode with CUSP or PCF (COMPAT) option
- ▶ Cryptographic domain number (DOMAIN) option
- ▶ Number of trace entries (TRACEENTRY) option
- ▶ CKDS name (CKDSN) option
- ▶ Maximum length for data in a callable service (MAXLEN) option
- ▶ Compatibility encryption algorithm (COMPENC) option
- ▶ User parameter (USERPARM) option
- ▶ PKDS name (PKDSN) option

### **ICSF Status Change (subtype 3)**

ICSF writes to subtype 3 when processors are verified at initialization, after a Master Key is set or changed, when ICSF switches from stand-by mode to normal mode, or when a processor comes online or offline. When processor status changes, subtype 3 gives the status of the processors still online. Subtype 3 contains the following information:

- ▶ Processor number
- ▶ Coprocessor number
- ▶ Cryptographic domain number
- ▶ Master Key version number

If a Master Key change or set occurs, subtype 3 also contains the following information:

- ▶ Master Key verification pattern
- ▶ Old Master Key verification pattern, if an old Master Key exists
- ▶ New Master Key verification pattern, if a new Master Key exists

### **Error Handling for Cryptographic Coprocessor Feature (subtype 4)**

ICSF writes to subtype 4 when the Coprocessor is in standby mode or when the Cryptographic Coprocessor Feature detects tampering. Subtype 4 contains the following information:

- ▶ Status word from the Cryptographic Coprocessor Feature
- ▶ Processor number
- ▶ Cryptographic domain number

### **Special Secure Mode Change (subtype 5)**

Subtype 5 contains special secure mode status bit. ICSF writes to subtype 5 when the status of special secure mode changes. ICSF also updates subtype 5 when the Cryptographic Coprocessor Feature indicates that special secure mode was required for an instruction, but was not enabled.

### **Master Key Part Entry (subtype 6)**

ICSF writes to subtype 6 when Master Key parts are entered using TKE workstation and are processed using the TKE Master Key entry ICSF panels. Subtype 6 contains the following information:

- ▶ The verification pattern for the Master Key part
- ▶ A bit indicating whether the verification pattern is valid
- ▶ The Coprocessor number
- ▶ The cryptographic domain number

If you enter the final Master Key part, the record also contains the verification pattern for the entire Master Key and a bit indicating whether the verification pattern is valid.

### **Operation Key Part Entry (subtype 7)**

ICSF writes to subtype 7 when key parts are entered using the TKE Workstation and are processed using the operational key entry ICSF panels. Subtype 7 contains the following information:

- ▶ The verification pattern for the key part
- ▶ A bit indicating whether the verification pattern is valid
- ▶ The cryptographic domain number
- ▶ The Coprocessor number
- ▶ The name of the CKDS that contains the entry with the key part
- ▶ The label of the CKDS entry that contains the key part

### **CKDS Refresh (subtype 8)**

ICSF writes to subtype 8 when the in-storage CKDS is successfully refreshed. ICSF refreshes the in-storage CKDS by reading a disk copy of a CKDS into storage. Subtype 8 contains the following information:

- ▶ Name of the current in-storage CKDS that ICSF refreshes
- ▶ Name of the disk copy of the CKDS that ICSF read into storage to replace the current CKDS

### **Dynamic CKDS Update (subtype 9)**

ICSF writes to subtype 9 when an application uses the dynamic CKDS update services to write to the CKDS. Subtype 9 contains the following information:

- ▶ Name of the changed CKDS
- ▶ The operation performed
- ▶ The CKDS entry (which includes the label name and key type) that was changed

### **PKA Key Part Entry (subtype 10)**

ICSF writes to subtype 10 when you use the ICSF panels to enter PKA master key parts. Subtype 10 contains the following information:

- ▶ An indication of which PKA Master Key is changing; the Signature Master Key (SMK), or the Key Management Master Key (KMMK)
- ▶ An indication of whether the following hash pattern of the PKA Master Key register is valid (it is valid when the final key part is entered).
- ▶ The hash pattern (MDC-4) of the PKA Master Key register
- ▶ The hash pattern of PKA key part
- ▶ The Coprocessor number
- ▶ Current cryptographic domain

If no DES Master Key has been validated, the key part entries do not contain a hash pattern. The record for the final key contains the hash pattern of the complete key.

### **Clear New Master Key Part Entry (subtype 11)**

ICSF writes to subtype 11 when you use the ICSF panels to enter new Master Key parts. Subtype 11 contains the following information:

- ▶ An indication of whether the following hash pattern of the new Master Key register is valid (it is valid when the final key part is entered).
- ▶ An indication of whether the following verification pattern of the new Master Key register is valid (it is valid when the final key part is entered).
- ▶ The hash pattern of the new Master Key register
- ▶ The verification pattern of the new Master Key register
- ▶ The hash pattern of new Master Key part
- ▶ The verification pattern of new Master Key part
- ▶ The Coprocessor number
- ▶ Current cryptographic domain

If no DES Master Key has been validated, the key part entries do not contain a verification pattern and hash pattern. The record for the final key contains the verification pattern and hash pattern of the complete key.

### **PKSC commands (subtype 12)**

ICSF writes to subtype 12 for every PKSC command entered through the CSFPKSC interface. Subtype 12 contains the following information:

- ▶ The complete PKSC request
- ▶ The corresponding PKSC response

### **Dynamic PKDS Update (subtype 13)**

ICSF writes to subtype 13 when an application uses the dynamic PKDS update services to change the PKDS. Subtype 13 contains the following information:

- ▶ The name of the changed PKDS
- ▶ The operation performed
- ▶ The name of the changed entry in the PKDS

### **PCI Cryptographic Coprocessor Clear Master Key Entry (subtype 14)**

ICSF writes to subtype 14 whenever you use ICSF panels to update SYM-MK and ASYM-MK in the new Master Key register in a PCI Cryptographic Coprocessor. Subtype 14 contains the following information:

- ▶ The Master Key (SYM-MK or ASYM-MK; NMK or key part) valid indicator
- ▶ The new Master Key verification pattern
- ▶ The key part verification pattern
- ▶ The PCI Cryptographic Coprocessor processor number
- ▶ The PCI Cryptographic Coprocessor serial number
- ▶ The domain index

### **PCI Cryptographic Coprocessor Retained Key Create or Delete (subtype 15)**

ICSF writes to subtype 15 whenever you create or delete a retained private key in a PCI Cryptographic Coprocessor. Subtype 15 contains the following information:

- ▶ The operation performed (created, deleted from PCI, deleted from PKDS)
- ▶ The retained key label
- ▶ The PCI Cryptographic Coprocessor processor number
- ▶ The PCI Cryptographic Coprocessor serial number
- ▶ The domain index

### **PCI Cryptographic Coprocessor TKE Command Request or Reply (subtype 16)**

ICSF writes to subtype 16 whenever a TKE Workstation either issues a command request to a PCI Cryptographic Coprocessor or receives a reply response from a PCI Cryptographic Coprocessor. Subtype 16 contains the following information:

- ▶ The indicator for request or reply
- ▶ The PCI Cryptographic Coprocessor processor number
- ▶ The PCI Cryptographic Coprocessor serial number
- ▶ The PCI Cryptographic Coprocessor domain index

- ▶ The request command block or reply response block length
- ▶ The request command data block or reply response data block length
- ▶ The request or reply CPRB

### PCI Cryptographic Coprocessor Timing (subtype 17)

ICSF periodically records processing times for PCI Cryptographic Coprocessor operations in subtype 17. Subtype 17 contains the following information:

- ▶ The time immediately before the operation begins
- ▶ The time immediately after the operation ends
- ▶ The time immediately after the results of the operation have been communicated to the caller address space
- ▶ The number of processes waiting to submit work to the same PCI Cryptographic Coprocessor, domain, and reference slot used by this operation
- ▶ The function code for this operation
- ▶ The PCI Cryptographic Coprocessor processor number
- ▶ The PCI Cryptographic Coprocessor serial number
- ▶ The PCI Cryptographic Coprocessor domain
- ▶ A reference number that identifies an internal ICSF queue element

### PCI Cryptographic Coprocessor Configuration (subtype 18)

ICSF writes subtype 18 when a PCI Cryptographic Coprocessor is brought online or taken offline. Subtype 18 contains the following information:

- ▶ The operation performed (PCI brought online or taken offline)
- ▶ The PCI Cryptographic Coprocessor processor number
- ▶ The PCI Cryptographic Coprocessor serial number

Figure 6-1 shows the output produced by the CSFSMFJ and CSFSMFR sample jobs.

```

Subtype=0012 PCI Cryptographic Coprocessor Configuration
Written when a PCI Cryptographic Coprocessor comes online or offline
12 Sep 2002 15:15:21.18
TME@... 0053CD96 DTE... 0102255F SID... VSN9      SSI... 00000000 STY...
0012
CGB... 80000000 CGX... 00 CGS... 92E01846
CGB 80 PCI CC brought online

```

Figure 6-1 Record type 82 edition

## 6.2.2 SMF record type 70, subtype 2

This record contains measurement data for PCI cryptographic coprocessors and accelerators. Record type 70 is written for each measurement interval and when the session terminates. It has the following sections:

### Cryptographic coprocessor data section

The fields in this section provide, for the specified PCICC and for the whole system; that is, all LPARs altogether:

- ▶ Execution time of all operations on the specified PCICC.
- ▶ Number of all operations on the specified PCICC.
- ▶ Number of all RSA-key-generation operations.

### Cryptographic accelerator data section

The fields in this section provide, for the specified PCICA and for each of its five engines, for the whole system:

- ▶ Execution time for all operations in 1024-bit-ME format (ME stands for “Modulus-Exponent”)
- ▶ Number of all operations in 1024-bit-ME format.
- ▶ Execution time for all operations in 2048-bit-ME format.
- ▶ Number of all operations in 2048-bit-ME format.
- ▶ Execution time for all operations in 1024-bit-CRT format. (CRT stands for “Chinese Remainder Theorem”)
- ▶ Number of all operations in 1024-bit-CRT format.
- ▶ Execution time for all operations in 2048-bit-CRT format.
- ▶ Number of all operations in 2048-bit-CRT format.

### Cryptographic Coprocessor Facility (CCF) data section

The fields in this section provide the following information for the set of active coprocessors

- ▶ Single DES:
  - Number of calls to encipher the data.
  - Number of bytes of data enciphered.
  - Number of instructions used to encipher the data.
- ▶ Triple DES:
  - Number of calls to encipher the data.
  - Number of bytes of data enciphered.
  - Number of instructions used to encipher the data.
- ▶ Single DES:
  - Number of calls to decipher the data.
  - Number of bytes of data deciphered.
  - Number of instructions used to decipher the data.

- ▶ Triple DES:
  - Number of calls to decipher the data.
  - Number of bytes of data deciphered.
  - Number of instructions used to decipher the data.
- ▶ MAC Generate:
  - Number of calls to generate the message authentication code.
  - Number of instructions used to generate the MAC.
- ▶ MAC Verify:
  - Number of calls to verify the MAC.
  - Number of bytes of data to verify the MAC.
  - Number of instructions used to verify the MAC.
- ▶ Hash:
  - Number of calls to hash the data.
  - Number of bytes of data hashed.
  - Number of instructions used to hash data.
- ▶ PIN Translate:
  - Number of calls to translate the PIN
- ▶ PIN Verify:
  - Number of calls to verify the PIN.

### 6.2.3 SMF record type 72, subtype 3

This record is written for a system running in WLM goal mode for each service or report class defined in the active service policy. It includes the goals and the actual measured values for this class. In the Service/Report Class Period Data section, information is given to the delays encountered at the cryptographic coprocessors utilization level. The fields in the section specifically report the following:

- ▶ Crypto FQ delay: A TCB was found that waits to be dispatched on a processor that has a CCF attached. This can happen when a CPU wants to execute a synchronous crypto instruction but has no direct connection to it.
- ▶ CAM crypto using samples: a TCB was found executing on a cryptographic asynchronous message processor.
- ▶ CAM crypto delay samples: a TCB was found waiting for a cryptographic asynchronous message processor.
- ▶ AP crypto using samples: a TCB was found executing on a cryptographic assist processor
- ▶ AP crypto delay samples: a TCB was found waiting for a cryptographic assist processor.

Refer to Figure 6-2 on page 252 for the output of cryptographic workload.

**Note:** In the terminology used here, Cryptographic Asynchronous Message Processor (CAM) designates PKA service requests routed to the CCF and Adjunct Processor (AP) designates service requests handled by the PCI coprocessors.

## 6.3 RMF reporting

With the availability of the new PCI cryptographic hardware (PCI Cryptographic Coprocessors and PCI Cryptographic Accelerators) on an LPAR basis, RMF provides performance monitoring with the Postprocessor Crypto Hardware Activity report which is based on SMF records type 70 subtype 2. These records are gathered by the Monitor I gathering option CRYPTO.

By default, this option activates data gathering. To suppress data gathering, you have to change CRYPTO to NOCRYPTO in ERBRMF00, or you have to add NOCRYPTO to your customized Parmlib member for Monitor I.

### Based on the SMF records type 70-2

The RMF Postprocessor can deliver reports displaying the following performance data:

- ▶ PCICC total rate
- ▶ PCICC total utilization
- ▶ PCICC total avg execution time
- ▶ PCICC key-gen rate
- ▶ PCICA 1024bit-ME rate
- ▶ PCICA 1024bit-ME utilization
- ▶ PCICA 1024bit-ME avg execution time
- ▶ PCICA 2048bit-ME rate
- ▶ PCICA 2048bit-ME utilization
- ▶ PCICA 2048bit-ME avg execution time
- ▶ PCICA 1024bit-CRT rate
- ▶ PCICA 1024bit-CRT utilization
- ▶ PCICA 1024bit-CRT avg execution time
- ▶ PCICA 2048bit-CRT rate
- ▶ PCICA 2048bit-CRT utilization
- ▶ PCICA 2048bit-CRT avg execution time

Figure 6-2 lists the output of the RMF postprocessor regarding cryptographic workload activity; in this case, it shows light SSL activity occurring in our test system.

```

1                                CRYPTO HARDWARE ACTIVITY

PAGE    1
      z/OS V1R3                SYSTEM ID VSN9                DATE 09/11/2002
INTERVAL 29.59.999                                RPT VERSION V1R2 RMF                TIME 14.00.00

CYCLE 1.000 SECONDS
0--- PCI CRYPTOGRAPHIC COPROCESSOR ---
----- TOTAL ----- KEY-GEN
ID  RATE  EXEC TIME  UTIL%  RATE
0 0  0.00   0.0    0.0    0.00
 1  0.00   0.0    0.0    0.00
-
----- PCI CRYPTOGRAPHIC ACCELERATOR
----- TOTAL ----- ME(1024) ----- ME(2048) -----
CRT(1024) ----- CRT(2048) -----
ID  RATE  EXEC TIME  UTIL%  RATE  EXEC TIME  UTIL%  RATE  EXEC TIME  UTIL%  RATE
EXEC TIME  UTIL%  RATE  EXEC TIME  UTIL%
0 2  0.18   1.3    0.0    0.00   0.0    0.0    0.00   0.0    0.0    0.18
1.3  0.0    0.00   0.0    0.0    0.0
 3  0.00   0.0    0.0    0.00   0.0    0.0    0.0    0.00   0.0    0.0    0.00
0.0  0.0    0.00   0.0    0.0
-
----- CRYPTOGRAPHIC COPROCESSOR FACILITY
-----
          DES ENCRYPTION    DES DECRYPTION    ----- MAC -----    - HASH -    ----- PIN
-----
          SINGLE  TRIPLE    SINGLE  TRIPLE    GENERATE  VERIFY                    TRANSLATE
VERIFIY
RATE      0.00   1.05    0.35   1.05    0.00   0.00    0.00    0.00
0.00
SIZE      0.00  209.3   176.0  297.3    0.00   0.00    0.00

```

Figure 6-2 RMF postprocessor output for cryptographic workload

### Based on SMF records type 72-3

- ▶ CPU Using %
- ▶ CPU Delay %
- ▶ Crypto Using %
- ▶ Crypto CAM Using %
- ▶ Crypto AP Using %
- ▶ Crypto Delay %
- ▶ Crypto CAM Delay %

- ▶ Crypto AP Delay %
- ▶ Crypto FQ Delay %

**Note:** In the terminology used here, Cryptographic Asynchronous Message Processor (CAM) designates PKA service requests routed to the CCF, and Adjunct Processor (AP) designates service requests handled by the PCI coprocessors.

Archived

Archived



## Linux for zSeries support of cryptographic coprocessors

In this chapter, we describe how Linux for zSeries supports the exploitation of the PCICC and PCICA cryptographic coprocessors.

## 7.1 Support of hardware coprocessors

Linux of zSeries does not support the CCF coprocessors. Instead, a generic device driver, z90crypt, is provided to route the cryptographic work to the PCICC or PCICA cards, as shown in Figure 7-1 on page 257.

According to Linux concepts, z90crypt is a device which is driven through the device node `/dev/z90crypt` using the device driver `z90crypt.o`. As such, z90crypt is invoked via the Linux I/O interface calls: get a device handle, open, read, ioctl, and close. As an example, “read” is used to get pseudo random bytes from the coprocessor, and other cryptographic services are requested via the “ioctl” function parameter.

In z90crypt, the focus is given to RSA cryptographic operations, the intent being mostly to provide hardware assistance to the SSL handshake. The extent of the hardware assistance depends on the type of PCI card used, as indicated in “The provided hardware services” on page 258, and all these cryptographic functions are performed using clear keys only.

Therefore, the support of the hardware cryptographic coprocessors support by Linux can be characterized as follows:

- ▶ If Linux for zSeries is the only hardware coprocessors exploiter running in the whole physical system, the CCF coprocessors do not have to be enabled (and therefore a system Power-on Reset is not required as a preamble to providing hardware crypto services).
- ▶ If PCICC cards are to be installed, the PCICC FCV diskette must have been imported and loaded into the HSA for proper initialization of the PCICC card(s) at installation time.

**Note:** If there is a mix of PCICC and PCICA in the system, z90crypt will use the PCICA card(s) only.

- ▶ .There is no hardware assistance provided for symmetric encryption and decryption as it is performed, for instance, during the data transfer part of the SSL protocol.
- ▶ Because the provided services use clear keys only, note the following:
  - No key store facility, or PKDS equivalent, is provided.
  - The crypto “domain” concept, although still applied for the PR/SM setup, is irrelevant to the exploitation of the hardware coprocessors by Linux for zSeries.

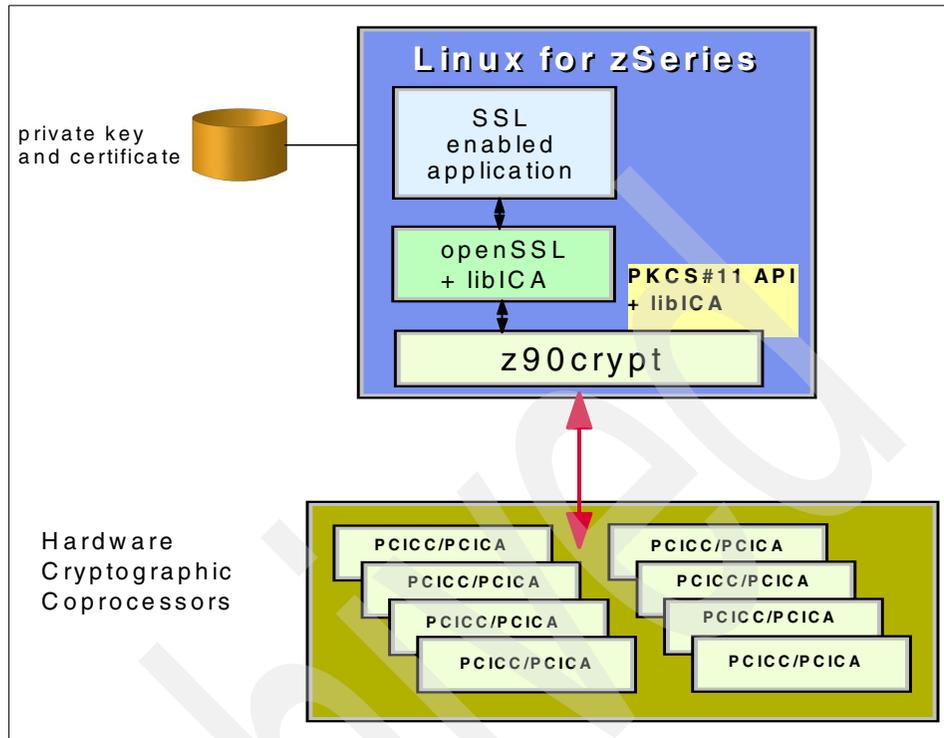


Figure 7-1 The z90crypt generic driver

Note that the z90crypt driver can still access the cryptographic coprocessors through virtualization layers, as shown in Figure 7-2 on page 258.

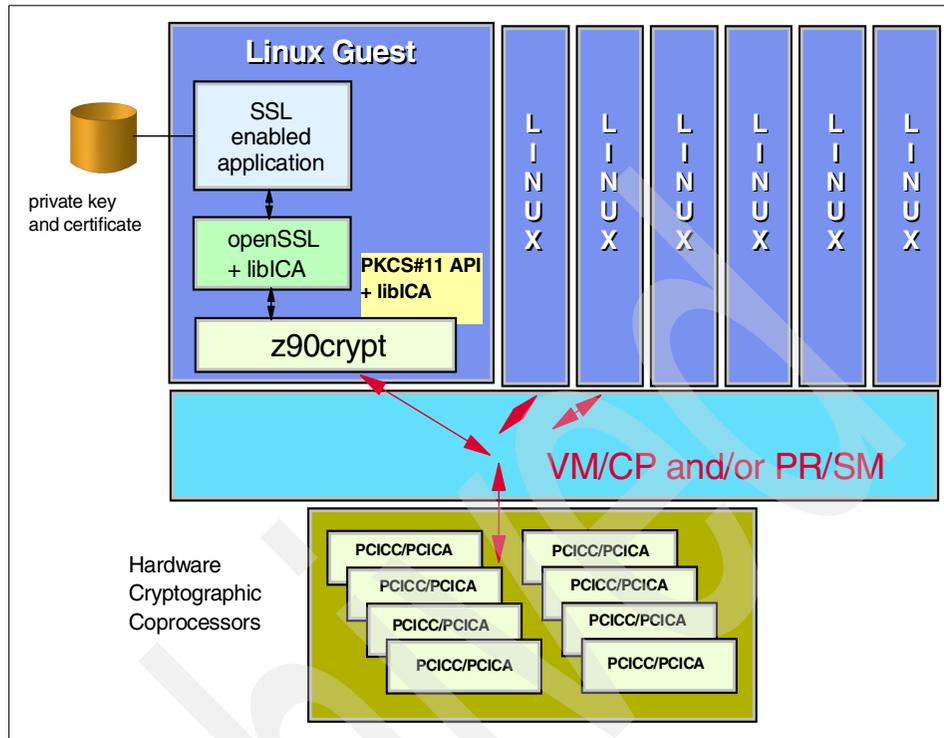


Figure 7-2 z90crypt and virtualization

## 7.1.1 The provided hardware services

The hardware services provided depend on the PCI card type used by z90crypt:

- ▶ If PCICC cards are used, the following operations can be performed by the coprocessors:
  - PKCS#1.2 RSA Public Key Encrypt (using clear key)
  - PKCS#1.2 RSA Private Key Decrypt (using clear key)
  - Pseudo Random Number Generation

PKCS#1.2 is the format used to transfer, during an SSL handshake, the “pre-master secret” value from the SSL client to the SSL server. For a detailed description of the RSA PKCS#1.2 standard, refer to:

<http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/>

Considering that this “pre-master secret” value is a plain-text message that must be securely transmitted, the application of the PKCS#1.2 consists of the following four steps:

- a. Encoding – also termed “padding”, which adds material to a plain-text message. In the case of the SSL handshake, this occurs in the SSL client.
- b. Encryption - with an RSA public key in the case of the SSL handshake (the SSL server’s public key). For an SSL handshake, this still occurs in the SSL client.
- c. Decryption - with an RSA private key in the case of the SSL handshake (the SSL server’s private key). This operation is performed by the SSL server.
- d. Decoding - for PKCS 1.2, this means stripping the padding from the message. This operation is performed by the SSL server.

So, if z90crypt provides PCICC support to an SSL server during the handshake phase, the last two steps are performed by the hardware.

- ▶ If PCICA cards are used, the following hardware services are provided (still using clear keys only):
  - RSA Public Key Encryption
  - RSA Public Key Decryption
  - RSA Private Key Encryption
  - RSA Private Key Decryption
  - Pseudo Random Number generation

Note that if the PKCS#1.2 format is to be used, the encoding and padding-stripping steps have to be performed by the requesting application. On the other hand, the mentioned hardware services can be conveniently used for RSA signature hardware generation and verification.

## 7.2 Access to cryptographic services

Linux for zSeries applications are not expected to call z90crypt directly; instead, they would look for cryptographic services as provided by higher-level APIs, such as:

- ▶ The PKCS#11 API. Actually implemented in the “openCryptoki” project, it is downloadable from:  
<http://www-124.ibm.com/developerworks/>
- ▶ The openssl “engine” with the ibmca crypto adapter interface; it is downloadable from:  
<http://www.openssl.org/>

Both of these APIs invoke cryptographic services from the libica shared library which, in turn, invoke the z90crypt driver to perform the hardware operations.

## 7.2.1 Functions of z90crypt API

As indicated, z90crypt drives only PCICC or PCICA coprocessors, and exploits the PCICA only if both PCI card types are installed in the system. It uses clear keys and provides a small set of functions through its own API, as described in the following sections.

### **Checking hardware status**

There is an ioctl interface for checking on underlying hardware in z90crypt. The function code is ICAZ90STATUS, and you supply a structure `ica_z90_status_t` (see header file “z90crypt.h” for the definition) as the argument. When control returns, you will have the number of cards installed and a mask showing which cards are installed; for example:

```
rc=ioctl(dh,ICAZ90STATUS,my_z90crypt_status);
```

where:

*rc* is your name for the ioctl return code.

*dh* is your name for the z90crypt device handle.

*my\_z90crypt\_status* is your name for your structure of type `ica_z90_status`, in which the status data is to be written.

**Note:** z90crypt, when initializing, provides the hardware status in a displayable form in the `/proc/driver/z90crypt` pseudo directory, as shown at “Checking the device status” on page 269.

### **Random number generation**

If you do a read from z90crypt, you will get back a string of pseudo random bytes. For read, you cannot specify a buffer length of more than 256 bytes; for example:

```
rc =read (dh,my_buffer,number_of_bytes);
```

where:

*rc* is your name for the ioctl return code.

*dh* is your name for the z90crypt device handle.

*my\_buffer* is your name for the byte-array where the random bytes will be loaded in response to the read.

### **Modular or CRT exponentiation (decryption)**

The functions are ICARSAMODEXPO and ICARSACRT. They correspond to the CSNDPKD service provided by ICSF in z/OS; that is, decryption of PKCS#1.2 formatted data using an RSA private key, depending on whether the clear private key is presented in the modulus-exponent format or in the CRT format.

The following steps are required in a decryption program:

1. Get a device handle, for example “dh”, for z90crypt:  
dh=open("/dev/z90crypt",O\_RDWR)
2. Create and load a structure of type `ica_rsa_modexpo_t` or `ica_rsa_modexpo_crt_t` as described below. You will define the private key to be used and set the input buffer pointer to the message you want decrypted.
3. Invoke `ioctl` to activate z90crypt:  
rc=ioctl(dh,<function code>,<structure name>)  
where:  
<function code> is ICARSAMODEXPO or ICARSACRT.  
<structure name> is the name of the structure you created, of type `ica_rsa_modexpo_t` or `ica_rsa_modexpo_crt_t`  
rc is your name for the return code.  
  
For example:  
rc=ioctl(dh,ICARSAMODEXPO,mycryptmex)  
where:  
rc is your name for the ioctl return code.  
dh is your name for the z90crypt device handle.
4. Obtain the decrypted and decoded message from the output buffer in the structure. The original message must have been PKCS 1.2-encoded; that is, the decrypted message must have correct padding. If it does have correct padding, z90crypt strips the padding; if it does not have correct padding, it gives an error message.

### **The `ica_rsa_modexpo_t` structure**

The `ica_rsa_modexpo_t` structure can be found in z90crypt header file “z90crypt.h”, along with the other structures for use with z90crypt. The (private) key consists of the exponent in `*b_key` and the modulus in `*n_modulus`. Both of these are hexadecimal representations of large numbers. The length L of `*n_modulus` must be in the range 64–256.

The input data and the exponent `b_key` must both be of length L. The output data must be of length L or greater. In all these cases, padding on the left with zeroes is allowed.

### ***The ica\_rsa\_modexpo\_crt\_t structure***

The only formal difference between this structure and the previous one is in the definition of the key. This is defined so that the Chinese Remainder Theorem can be used in decryption/encryption. z90crypt decrypts about four times faster if the CRT definition is used. The key-definition fields are all in hexadecimal representation. They have these meanings and limitations:

- ▶ `*bp_key`, `*bq_key`: Prime factors of the modulus.  
In z90crypt, the modulus is  $(*bp\_key) * (*bq\_key)$ . The resulting length  $L$  of the modulus, in hexadecimal representation, must be found before these fields are defined.
- ▶ `*np_prime`, `*nq_prime`: Exponents used for `*bp_key` and `*bq_key`, respectively.
- ▶ `*u_mult_inv`: A coefficient used in the z90crypt implementation of decryption by CRT.
- ▶ `*bp_key`, `*np_prime`, `*u_mult_inv` must all be of length  $8 + L/2$ .
- ▶ `*bq_key`, `*nq_prime` must both be of length  $L/2$ .

The input data length must be  $L$ , and the output data length must be at least  $L$ , as in:

```
ca_rsa_modexpo_t
```

### ***Quiescing***

You need root authority to perform quiescing. You can “quiesce” z90crypt by executing an `ioctl` with function code `ICAZ90QUIESCE`. This lets the driver finish any outstanding work, but prevents any new work from being submitted.

After 30 seconds, either all outstanding requests will be completed, or else the requesting process will be notified that its work will not be completed.

The only way to “un-quiesce” z90crypt is to unload it and then reload it; for example:

```
rc =ioctl(dh,ICAZ90QUIESCE)
```

### ***Returns from read and ioctl***

A return code of 0 means everything went well and the result is in your output buffer. Return codes greater than 0 have the following meanings:

<b>8</b>	Invalid operand
<b>16</b>	Device not available
<b>17</b>	Error in pkcs 1.2 padding (no room for required padding)
<b>24</b>	Error copying to or from user space

A return code of -1 means that errno is one of the following:

<b>13</b>	Permission denied (attempted quiesce, but not root)
<b>22</b>	Invalid operation
<b>132</b>	Device is quiescing; no new work being accepted
<b>134</b>	Unknown error (usually means a transient error in a crypto device; a retry may succeed)

For information on any other error code, look in `/usr/include/asm/errno.h`.

## 7.2.2 The libica API

You can download libica from:

<http://oss.software.ibm.com>

The libica API offers the following cryptographic functions:

- ▶ `icaGetAdapterMask`
- ▶ `icaGetAdapterID`
- ▶ `icaGetVPD`
- ▶ `icaOpenAdapter`
- ▶ `icaCloseAdapter`
- ▶ `icaRsaModExpo`
- ▶ `icaRsaCrt`
- ▶ `icaRsaKeyGenerateModExpo`
- ▶ `icaRsaKeyGenerateCrt`
- ▶ `icaRsaKeyReGenerateModExpo`
- ▶ `icaRsaKeyReGenerateCrt`
- ▶ `icaDesEncrypt`
- ▶ `icaDesDecrypt`
- ▶ `icaTDesEncrypt`
- ▶ `icaTDesDecrypt`
- ▶ `icaDesMac`
- ▶ `icaTDesMac`
- ▶ `icaSha1`
- ▶ `icaRandomNumberGenerate`

The libica API makes the open, read, ioctl, and close calls to z90crypt when appropriate, that is, to get the hardware provided services described in “The provided hardware services” on page 258.

## 7.2.3 The PKCS#11 API

The openCryptoki project's implementation of the PKCS#11 API is downloadable from:

<http://oss.software.ibm.com>

For the specification of PKCS#11, see:

<http://rsasecurity.com/rsalabs/pkcs>

The openCryptoki code loads libica (see “The libica API” on page 263) and uses it for all cryptographic operations.

## 7.2.4 Functions of the OpenSSL “engine”

OpenSSL provides an SSL “engine” that can access libica for RSA operations after the IBM patch from the libica project at developerWorks™ is applied; see:

<http://openssl.org>

## 7.3 Virtualization

In the following section, we describe virtualization.

### LPAR definition

1. With the PCICC and/or PCICA card(s) installed, open your Linux LPAR image profile. Select the panel with the **Processor** tab. This offers the choice of the two built-in Cryptographic Co-processor Features (CCFs). One or both must be chosen for carrying on the PCICC/PCICA card configuration of your LPAR, as the CCF does not operate under Linux. You must click on one or both, but it makes no difference which you choose.
2. Select a unique cryptographic domain for the LPAR.
3. When the CCF(s) are selected, two further tabs appear to the right: the “crypto” tab and the “PCI crypto” tab; do the following:
  - a. Select the panel with the “crypto” tab and assign a unique usage domain index to your partition. Make the control domain index the same as the usage domain index.

The index is a number between 0 and 15. The cryptographic domain must be unique to the LPAR— different from the cryptographic domain for any other LPAR. (The choice, however, has no visible effect since the PCICC will be operated using clear keys.)
  - b. Select the panel with “PCI Crypto”. This offers a choice of PCICCs and/or PCICAs cards to be made available to the partition, and to be brought automatically online at partition activation. (Be aware, however, that z90crypt uses only PCICAs, when both types of adapters are installed.)
4. If you are using a zSeries z900 GA2 machine, do the following:

On the PR/SM panel where you choose which architecture your LPAR will support, choose **ESA390**. (Do *not* choose “Linux only”; if you choose Linux only, no crypto devices will be available to your LPAR.)

### 7.3.1 Using a crypto device in VM

VM/CP provides VM guest machines with access to the real cryptographic coprocessors. Note that it does not provide simulation of hardware cryptographic coprocessors (that is, real coprocessors must be operating in the system). VM/CP intercepts the crypto instructions issued by the guest machines and manages the request queues for the guests sharing the coprocessors.

Giving access to the cryptographic coprocessors to a VM guest machine is actually a two-step process:

1. The guest machine must be permitted to use the crypto coprocessors in its entry of the VM directory; refer to “CRYPTO directory control statement” on page 265.
2. The crypto device must be defined in the operating system running in the guest machine. For Linux for zSeries, this consists of:
  - Creating the device node
  - Loading and activating the device driver; refer to “Installing the crypto device driver” on page 266.

#### **CRYPTO directory control statement**

The CRYPTO directory control statement allows a virtual machine to use the hardware cryptographic coprocessors. To use the PCICC or PCICA, the new APVIRT operand should be specified. (for example, CRYPTO APVIRT).

Note that the CRYPTO statement, if included in the directory entry, must appear *after* the USER statement and *before* any device statements. The APVIRT operand tells CP that this virtual machine may use the clear-key functions of the Adjunct Processor cryptographic facility.

You do not need to specify the DOMAIN or CSU (that is, CCF ID) operands when specifying the APVIRT operand. (Actually, you do not want to specify a DOMAIN number, as each guest would then require a unique number in the range 0 to 15.)

**Important:** Querying the crypto facility from VM maint, or from the guest VM, or by using the z90crypt status, may return a domain number. This is a virtual domain attributed by VM/CP which is irrelevant to the hardware crypto support as provided by Linux.

## 7.4 Our installation with a 31-bit Linux

In the following sections, we describe our installation tasks.

### 7.4.1 Preparation

We used the following distribution to test the exploitation of the PCICA card on a z900 system by Linux for zSeries: SuSE SLES-7 (s390) - Kernel 4.7-SuSE-SMP. All examples provided in this chapter refer to this distribution.

The following directories were created:

- ▶ /opt/z90crypt
- ▶ /opt/z90crypt/test
- ▶ /opt/z90crypt/test/icatest
- ▶ /opt/z90crypt/test/z90ctest

The Object Code Only (OCO) crypto device driver package for the Linux kernel 2.4.7 was downloaded from:

[www10.software.ibm.com/developerworks/opensource/linux390/index.shtml](http://www10.software.ibm.com/developerworks/opensource/linux390/index.shtml)

The z90crypt-2.4.7-s390-1.tar.gz file was unpacked into /opt/z90crypt.

### 7.4.2 Installing the crypto device driver

The install script provided with the package was not directly applicable to the SuSE sles7 (31-bit, 2.4.7 kernel) installed on our system. We first made a backup copy (z90crypt\_install.ORIG), then we edited it to make the appropriate changes.

The install script we used is shown in Figure 7-3 on page 267.

```

#!/bin/sh -
#*****/
#* */
#* s390 only */
#* */
#* Copyright (c) IBM Corporation 2001 */
#* Licensed Material - Program Property of IBM */
#* All rights reserved */
#* Licensed under the IBM Public License (IPL) */
#* */
#* Script to be used to install device driver z90crypt. */
#* */
#*****/
# mkdir -p /lib/modules/misc
# install -c z90crypt.o /lib/modules/misc
install -c z90crypt.o /lib/modules/2.4.7-SuSE-SMP/misc/
install -c z90crypt.h /usr/include/linux

```

Figure 7-3 z90crypt modified install script

A list of the files involved in the installation is shown in Figure 7-4.

```

linux548:/opt/z90crypt # ls -al
total 101
drwxr-xr-x  2 rootroot  328 Dec  4 10:26 .
drwxr-xr-x 12 root  root   296 Dec  4 10:03 ..
-rw-r--r--  1 root  root   619 Nov  9  2001 LICENSE
-rw-r--r--  1 root  root 12703 Nov  9  2001 LICENSE.IPL
-rw-r--r--  1 root  root  6432 Nov  9  2001 README
-rw-r--r--  1 root  root  7176 Nov  9  2001 z90crypt.h
-rw-r--r--  1 root  root 45904 Nov  9  2001 z90crypt.o
-rwxr-xr-x  1 root  root  1057 Dec  4 10:09 z90crypt_install
-rwxr-xr-x  1 root  root   997 Nov  9  2001 z90crypt_install.ORIG
-rwxr-xr-x  1 root  root  1344 Nov  9  2001 z90crypt_load
-rwxr-xr-x  1 root  root  1239 Nov  9  2001 z90crypt_unload

```

Figure 7-4 z90crypt installation files

### 7.4.3 Running the modified install script

Executing the install script resulted in the installation of the z90crypt.o file in /lib/modules/2.4.7-SuSE-SMP/misc, as shown in Figure 7-5 on page 268.

```

linux548:/opt/z90crypt # ./z90crypt_install

linux548:/opt/z90crypt # ls -l /lib/modules/2.4.7-SuSE-SMP/misc/
total 109
drwxr-xr-x  2 root  root          128 Dec  4 10:28 .
drwxr-xr-x  6 root  root          416 Dec  3 14:04 ..
-rw-r--r--  1 root  root        8505 Oct 30 2001 cpint.o
-rw-r--r--  1 root  root       47188 Oct 30 2001 qdio.o
-rwxr-xr-x  1 root  root       45904 Dec  4 10:28 z90crypt.o

```

Figure 7-5 Running the modified install script

### 7.4.4 Loading the device driver and defining the crypto device node

Load/unload scripts are available with the package to perform these operations. We used the load script, with the results shown in Figure 7-6.

```

linux548:/opt/z90crypt # ./z90crypt_load
linux548:/opt/z90crypt # lsmod
Module                Size Used by
z90crypt             31584  0 (unused)
af_packet             16496  0 (autoclean)
ipv6                 257584 -1 (autoclean)
qeth                 135488  1 (autoclean)
qdio                 39504  1 (autoclean) [qeth]
ipchains             44656  0

```

Figure 7-6 z90crypt\_load execution

The load script itself is shown in Figure 7-7 on page 269.

```

#!/bin/sh -
#*****/
#*****/
module="z90crypt"
device="z90crypt"
group="root"
mode="666"
# invoke insmod with all arguments we got
/sbin/insmod -f -m $module $* >z90crypt.map || exit 1
major=`cat /proc/devices | awk "\$2==\"module\"{print \\$1}"`
# Remove stale nodes and replace them, then give gid and perms
# create unrouted device (minor b'00000000' decimal 0)
rm -f /dev/${device}
mknod /dev/${device} c $major 0
chgrp $group /dev/${device}
chmod $mode /dev/${device}

```

Figure 7-7 The z90crypt\_load script

### 7.4.5 Checking the device status

In this example, we first interrogate the status of the z90crypt driver which, during its startup, has queried the status of the installed hardware coprocessor(s). The hardware can be retrieved by displaying the pseudo directory /proc/driver/z90crypt.

This is depicted in Figure 7-8 on page 270, where we can see that the system we used had two PCICA coprocessors installed.

```

linux548:~ # rcz90crypt status
Checking for module z90crypt: z90crypt 4272 0 (unused) running
linux548:~ # cat /proc/driver/z90crypt
Cryptographic domain: 8
Total device count: 2
PCICA count: 2
PCICC count: 0
requestq count: 0
pendingq count: 0
Total open handles: 0
Mask of online devices: 01 means PCICA, 02 means PCICC
 0101000000000000 0000000000000000 0000000000000000 0000000000000000
 0000000000000000 0000000000000000 0000000000000000 0000000000000000
Mask of waiting work element counts
 0000000000000000 0000000000000000 0000000000000000 0000000000000000
 0000000000000000 0000000000000000 0000000000000000 0000000000000000
The system is using 2 PCI CA cards

```

Figure 7-8 Checking the device status

## 7.5 Low-level testing

Two small programs written in C were used to perform low-level tests using the z90crypt and libica APIs. These tools consist of the files shown in Figure 7-9.

The file content is shown in “Low-level test programs” on page 280 (except for the ica\_api.h file, which can be found in the libica package).

```

linux548:/opt/z90crypt/test/z90ctest # tar -xvzf ../testz90c.tgz
tstcrtde.c
tell.h
tellit.c
makcrtde
linux548:/opt/z90crypt/test/icatest # tar -xvzf ../testica.tgz
icacrtde.c
tellit.c
tell.h
ica_api.h
makicacr

```

Figure 7-9 Installation of the low-level test programs

## Low-level testing of the z90crypt device

A decryption test with a clear private key in CRT format, called tstcrtde, was installed and built, as shown in Figure 7-10.

```
linux548:/opt/z90crypt/test/z90ctest # make -f makcrtde
cc -DZ90CRYPT_DEBUG -D_REENTRANT -I. -O -g -Wall -c -o tstcrtde.o
tstcrtde.c
cc -DZ90CRYPT_DEBUG -D_REENTRANT -I. -O -g -Wall -c -o tellit.o
tellit.cgcc -o tstcrtde tstcrtde.o tellit.o
linux548:/opt/z90crypt/test/z90ctest # ls -al
total 109
drwxr-xr-x  2 root  root           232 Dec  4 10:56 .
drwxr-xr-x  4 root  root           160 Dec  4 10:54 ..
-rw-r--r--  1 linux548 500           232 Sep 13 20:32 makcrtde
-rw-r--r--  1 linux548 500           284 May  8 2002 tell.h
-rw-r--r--  1 linux548 500          7779 May  8 2002 tellit.c
-rw-r--r--  1 root  root          18460 Dec  4 10:56 tellit.o
-rwxr-xr-x  1 root  root           40131 Dec  4 10:56 tstcrtde
-rw-r--r--  1 linux548 500           4289 Sep  4 15:35 tstcrtde.c
-rw-r--r--  1 root  root          24476 Dec  4 10:56 tstcrtde.o
```

Figure 7-10 Compilation and build of the z90crypt low-level test

Then the low-level test was run, as shown in Figure 7-11.

```
linux548:/opt/z90crypt/test/z90ctest # ./tstcrtde
Open of z90crypt succeeded
rv returned by ioctl: 0
Output Data Length returned: 64
Output returned:
 0002a0a6c6823dcb 875a001122334455 6677889900112233 4455667788990011
2233445566778899 0011223344556677 8899001122334455 6677889900112233
Raw buffer returned
 7ffff6b000000040 7ffff6f000000040 7ffff6407ffff668 7ffff5f87ffff620
7ffff688
icacrt.outputdata:
 0002a0a6c6823dcb 875a001122334455 6677889900112233 4455667788990011
2233445566778899 0011223344556677 8899001122334455 6677889900112233
Correct result returned.
```

Figure 7-11 Low-level test of z90crypt device driver

## 7.5.1 Installation and test of the Crypto Interface Library (libica)

The IBM eServer Cryptographic Accelerator Device Interface Library (libica) for the Linux Operating System was downloaded from the IBM developer works projects at:

<http://oss.software.ibm.com/developerworks/opensource/libica>

The generalized Interface library for the IBM eServer Cryptographic Accelerator Device Driver is a low-level API for the specified adapter. It is not intended to be an interface that is written to by applications.

Applications should use the openCryptoki PKCS#11 API for interfacing to the token, or use a generic facility middleware engine such as OpenSSL for SSL-enabled applications.

We downloaded two sets of packages from this site:

- ▶ libica (either rpm or source tar)
- ▶ openssl patches

### libica

For compatibility with the SuSE sles7 environment installed on our system, we used the libica-1.1.3 source package to locally build the libica.so; it is available at developerWorks:

<http://oss.software.ibm.com/developerworks/opensource/libica>

### OpenSSL patches

The patches we used to implement the IBM Crypto Adapter support were contained in the openssl-Patches package at release Patch-release2. We used these files:

- ▶ Configure-390-dso.patch - needed to enable DSOs on system 390 and 390x for engine support
- ▶ ibmca.patch-0.96e-2 - ICA device engine patch

### Unpacking source packages for libica-1.1.3 and OpenSSL

The packages were unpacked into the ssl-web directory, as shown in Figure 7-12 on page 273.

```
linux548:/usr/local/src/ssl-web # ls -al
total 6
drwxr-xr-x  6 root    root      184 Dec  4 18:43 .
drwxr-xr-x  3 root    root      72 Dec  4 13:26 ..
drwxr-xr-x  8 1078   1078    520 Jun 18 20:22 apache_1.3.26
drwxrwxr-x  5 linux548 500     904 Dec  4 18:43 libica-1.1.3
drwxr-xr-x 10 root    root     680 Dec  4 13:28
mod_ssl-2.8.10-1.3.26d
rwxr-xr-x  20 root    root    1416 Dec  4 15:52 openssl-0.9.6e
```

Figure 7-12 The apache, libica, mod-ssl and openssl packages

### Building OpenSSL with ibmca engine included

The OPENSSL patches were applied to the source package for the following purposes:

1. To add ldl for s390 platform into Configure
2. To include the vendor crypto engine ibmca

The patch application is shown in Figure 7-12 on page 273.

```

linux548:/usr/local/src/ssl-web/openssl-0.9.6e # patch -p0 <
/home/linux548/TempXfer/Configure-390-dso.patch
(Stripping trailing CRs from patch.)
patching file Configure
Hunk #1 succeeded at 354 (offset 2 lines).
linux548:/usr/local/src/ssl-web/openssl-0.9.6e # patch -p1 <
/home/linux548/TempXfer/ibmca.patch-0.96e-2
patching file crypto/engine/Makefile
patching file crypto/engine/Makefile.ssl
patching file crypto/engine/engine.h
patching file crypto/engine/engine_err.c
patching file crypto/engine/engine_int.h
patching file crypto/engine/engine_list.c
patching file crypto/engine/hw_ibmca.c
patching file crypto/engine/vendor_defns/ica_openssl_api.h
linux548:/usr/local/src/ssl-web/openssl-0.9.6e # ./config
.....
linux548:/usr/local/src/ssl-web/openssl-0.9.6e # make
.....
linux548:/usr/local/src/ssl-web/openssl-0.9.6e # make test
.....
linux548:/usr/local/src/ssl-web/openssl-0.9.6e # make install

```

Figure 7-13 Installation of patch and build of OpenSSL

**Note:** The PREFIX option was not specified. The new, locally built openssl with ICA engine is by default installed under /usr/local/ssl ./bin ./lib, etc.

### Building and installing libica-1.1.3

We must first insure that the following SuSE sles7 packages are installed: autoconf, automake and openssl\_devel from group “d Development (C, C++, Lisp, etc.)”

The building of libica is shown in Figure 7-14 on page 275.

```

linux548:~ # cd /usr/local/src/ssl-web/libica-1.1.3/
linux548:/usr/local/src/ssl-web/libica-1.1.3 # ./configure
creating cache ./config.cache
checking host system type... s390-ibm-linux-gnu
checking target system type... s390-ibm-linux-gnu
checking build system type... s390-ibm-linux-gnu
checking for a BSD compatible install... ..
.....
linux548:/usr/local/src/ssl-web/libica-1.1.3 # make
Making all in src
make[1]: Entering directory `/usr/local/src/ssl-web/libica-1.1.3/src'
cd .. && automake --gnu src/Makefile
cd .. \
&& CONFIG_FILES=src/Makefile CONFIG_HEADERS= /bin/sh ./config.status
creating src/Makefile
make[1]: Leaving directory `/usr/local/src/ssl-web/libica-1.1.3/src'
make[1]: Entering directory `/usr/local/src/ssl-web/libica-1.1.3/src'
gcc -DPACKAGE="libica" -DVERSION="1.1.3" -DSTDC_HEADERS=1 -DH...
linux548:/usr/local/src/ssl-web/libica-1.1.3 # make install
Making install in src
make[1]: Entering directory `/usr/local/src/ssl-web/libica-1.1.3/src'
make[2]: Entering directory `/usr/local/src/ssl-web/libica-1.1.3/src'
/bin/sh ../mkinstalldirs /usr/lib
/usr/bin/install -c libica.so /usr/lib/libica.so
cp ../include/ica_api.h /usr/include
.....

```

Figure 7-14 Building and installation of libica

At this point, libica.so is installed in /usr/lib. We just created new shared libraries, so it is time to refresh the links and cache of commonly accessed shared libraries by entering:

```
linux548:/usr/local/src/ssl-web/libica-1.1.3 # ldconfig
```

### Low-level testing via ica api

We built the ica crypto test tool as indicated in Figure 7-15 on page 276.

```

linux548:~ # cd /opt/z90crypt/test/icatest/
linux548:/opt/z90crypt/test/icatest # ls
. .. ica_api.h icacrtdc icacrtdc.c makiacr tell.h tellit.c
linux548:/opt/z90crypt/test/icatest # make -f makiacr
cc -DZ90CRYPT_DEBUG -D_REENTRANT -O -g -Wall -c -o icacrtdc.o icacrtdc.c
cc -DZ90CRYPT_DEBUG -D_REENTRANT -O -g -Wall -c -o tellit.o tellit.c
gcc -licr -o icacrtdc icacrtdc.o tellit.o

```

Figure 7-15 Building the ica test

## Running the low-level ica test

This assumes that the z90crypt device driver is loaded and the z90crypt low-level test was successful, as run in “Low-level testing of the z90crypt device” on page 271.

The run of the low-level ica test is shown in Figure 7-16.

```

linux548:/opt/z90crypt/test/icatest # ./icacrtdc
Open of libica succeeded
rv returned by libica: 0
Output Data Length returned: 64

Output returned:
00028fd77ec6e0a1 24c948f229c87aeb dd1ec3e043e4b303 db40ac38da643b77
184da75a13fcd084 8050381fbffc5600 1122334455667788 1122334455667788
Correct result returned

```

Figure 7-16 Running the ica low-level test

## Testing with the OpenSSL engine

OpenSSL can be used to test the execution speed of the RSA signature in interactive mode. We used this facility to assess the speed difference when invoking the crypto coprocessors, as opposed to using a purely software engine. This test is shown in Figure 7-17 on page 277.

```

linux548:/usr/local/ssl/bin # ./openssl
OpenSSL> speed rsa512
Doing 512 bit private rsa's for 10s: 1424 512 bit private RSA's in 10.00s
Doing 512 bit public rsa's for 10s: 14104 512 bit public RSA's in 10.01s
OpenSSL 0.9.6e [engine] 30 Jul 2002
built on: Wed Dec  4 15:46:52 CET 2002
options:bn(64,32) md2(int) rc4(ptr,int) des(idx,cisc,4,long) idea(int)
blowfish(idx)compiler: gcc -fPIC -DTHREADS -D_REENTRANT -DDSO_DLFCN
-DHAVE_DLFCN_H -DB_ENDIAN -DTERMIO -DNO_ASM -O3 -fomit-frame-pointer -Wall
sign    verify    sign/s verify/s
rsa 512 bits  0.0070s  0.0007s  142.4   1409.0
OpenSSL> speed -engine ibmca rsa512
engine "ibmca" set.
Doing 512 bit private rsa's for 10s: 1000 512 bit private RSA's in 0.00s
Doing 512 bit public rsa's for 10s: 1000 512 bit public RSA's in 0.00s
OpenSSL 0.9.6e [engine] 30 Jul 2002
built on: Wed Dec  4 15:46:52 CET 2002
options:bn(64,32) md2(int) rc4(ptr,int) des(idx,cisc,4,long) idea(int)
blowfish(idx)
compiler: gcc -fPIC -DTHREADS -D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H
-DB_ENDIAN -DTERMIO -DNO_ASM -O3 -fomit-frame-pointer -Wall
sign    verify    sign/s verify/s
rsa 512 bits  0.0000s  0.0000s 1000000.0 1000000.0
OpenSSL> quit

```

Figure 7-17 Running the speed test of OpenSSL

## 7.6 Example SSL-enabled application: Apache Web server

The Apache HTTP server uses a mod\_ssl module to interact with the openssl engine in order to perform SSL communications, as shown in Figure 7-18 on page 278.

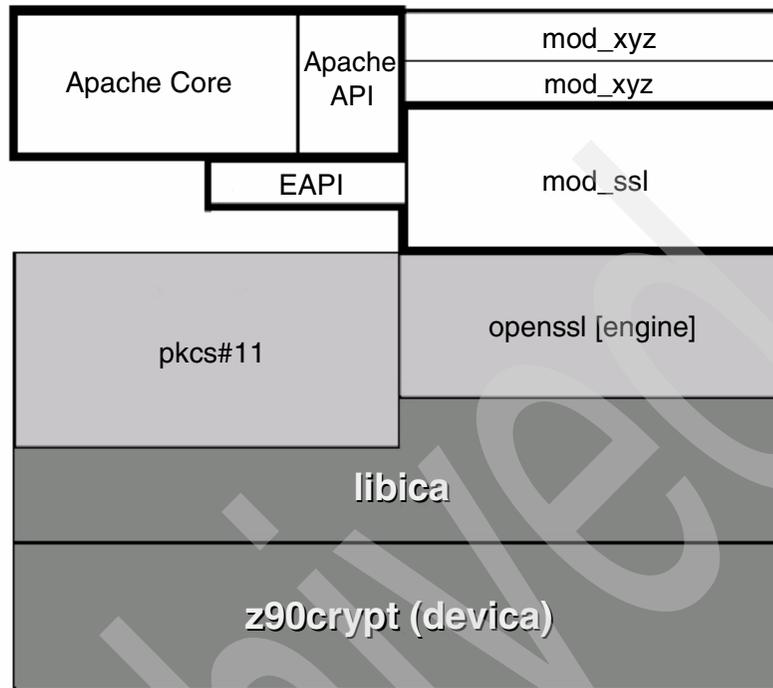


Figure 7-18 Apache HTTP server enablement for SSL

The Apache Web server and the mod\_ssl module have been downloaded into /usr/local/src/ssl\_web.

Now mod\_ssl must be configured to compile it into Apache with SSL and crypto device enabled. The mod\_ssl configure utility will use the selected configuration options to appropriately apply the patch in Apache source trees and create .apci in the Apache source directory.

We will be able to compile, build, and install the Apache Web server directly, without running apache configure again., just by proceeding as indicated in the last lines of the mod\_ssl configuration output messages.

The following options are required:

- ▶ Path to the Apache source directory
- ▶ Path to the open\_ssl engine that we previously compiled
- ▶ ssl enable flag
- ▶ Rule SSL\_EXPERIMENTAL to enable the vendor extension code. Use the SSLCryptoDevice configuration statement in httpd.conf to define the ibmca crypto device.

- ▶ The path prefix for installing the newly built Apache Web Server

Figure 7-19 shows the options we used.

```
lin64ux548:/usr/local/src/ssl-web/mod_ssl-2.8.10-1.3.26 # ./configure \  
> --with-apache=../apache_1.3.26 \  
> --with-ssl=/usr/local/ssl \  
> --enable-module=ssl \  
> --enable-rule=SSL_EXPERIMENTAL \  
> --prefix=/usr/local/apache
```

*Figure 7-19 Apache mod\_ssl configure utility*

At the end of the execution of the `configure` command, the last four messages directed us to execute the following, in sequence: `make`, `make certificate`, and `make install` to build and create a self-signed certificate, and then install the Apache Web Server. Real-time explanations are provided at the console when performing these steps.

Finally, the crypto device had to be defined in the `httpd.conf` file to provide access to the hardware cryptographic coprocessors, by adding a directive as shown in Figure 7-20.

```
<IfModule mod_ssl.c>  
SSLCryptoDevice ibmca
```

*Figure 7-20 Apache configuration file directive*

After restarting the Apache HTTP server with `apachectl startssl`, we verified that there was now one open handle showing in the `z90crypt` driver monitoring record in the `/proc` pseudo directory, as shown in Figure 7-21 on page 280.

```

lin64ux548:/usr/local/ssl/bin # cat /proc/driver/z90crypt
Cryptographic domain: 8
Total device count: 2
PCICA count: 2
PCICC count: 0
requestq count: 0
pendingq count: 0
Total open handles: 1
Mask of online devices: 01 means PCICA, 02 means PCICC
0101000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
Mask of waiting work element counts
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000

```

Figure 7-21 Checking the status of z90crypt

## 7.7 Low-level test programs

The following sections list the programs used to invoke coprocessor functions through the z90crypt and libica APIs.

### 7.7.1 testcrtde.c

This program calls z90crypt to perform a decryption using an RSA key token in the CRT format.

```

#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <linux/z90crypt.h>
#include "tell.h"
#define _POSIX_SOURCE 1
#include <unistd.h>
#include <errno.h>
int main ()
{
int dh;
int rv = 0;
/*

```

```

unsigned char pubexp[64] = {
0x2F,0x8E,0xD8,0x75,0x14,0xB6,0xE6,0x93,0x68,0x82,0xF6,0x6A,0xFA,0x83,0
xA2,0x4C,
0xE2,0x1C,0x54,0xE0,0xD4,0xE0,0xEF,0xC3,0x0E,0x32,0x10,0x96,0x3A,0x10,0
xBB,0x37,
0xCB,0x23,0x41,0x40,0xF8,0x2A,0x02,0x13,0x86,0xB4,0x92,0x5C,0xC7,0xD3,0
x3C,0x2C,
0xC6,0x77,0x69,0xD4,0x68,0xC4,0x4D,0x19,0xC0,0xC4,0xAA,0x7B,0xD2,0x3D,0
x29,0xB1
};
unsigned char mod[64] =
{0xCE,0x02,0x68,0x2D,0x5F,0xA9,0xDE,0x0C,0xF6,0xD2,0x7B,0x58,0x4B,0xF9,
0x28,0x68,
0x3D,0xB4,0xF4,0xEF,0x78,0xD5,0xBE,0x66,0x63,0x42,0xEF,0xF8,0xFD,0xA4,0
xF8,0xB0,
0x8E,0x29,0xC2,0xC9,0x2E,0xD8,0x45,0xB8,0x53,0x8C,0x6F,0x4E,0x72,0x8F,0
x6C,0x04,0x9C,0x88,0xFC,0x1E,0xC5,0x83,0x55,0x57,0xF7,0xDD,0xFD,0x4F,0x
11,0x36,0x95,0x5D
};
*/
unsigned char msg[64] = {
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x11,0x22,0x33,0
x44,0x55,
0x66,0x77,0x88,0x99,0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0
x00,0x11,
0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0x00,0x11,0x22,0x33,0x44,0x55,0
x66,0x77,
0x88,0x99,0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0x00,0x11,0
x22,0x33
};
unsigned char p[40] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xE8,0xBD,0x11,0x73,0xC8,0x0A,
0xC6,0x35,
0x53,0x77,0x05,0x0C,0x70,0x34,0xA5,0x70,0x40,0x59,0x50,0x98,0x90,0xC1,0
x82,0x10,
0xF7,0xD7,0x27,0x5F,0xE3,0x81,0x25,0xDF
};
unsigned char q[32] =
{0xE2,0x99,0x6F,0xF1,0xB6,0xEE,0x6A,0x61,0x77,0x4C,0xB3,0xD4,0xCF,0xF0,
0x39,0x46,
0xE8,0x0E,0x05,0xAF,0x10,0x05,0x73,0x54,0xA8,0x0F,0x36,0x6B,0xAC,0x93,0
xD4,0x43
};
unsigned char dp[40] = {

```

```

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xA3,0x49,0xD1,0x8A,0x23,0x0C,0
xB0,0x01,
0x8B,0xAD,0xFC,0x97,0xD0,0x63,0xB1,0xD0,0x82,0x4C,0x9C,0x46,0xA5,0x4B,0
x8C,0x8D,
0xBB,0x5B,0x9D,0x9A,0xE4,0xFE,0x2A,0xF3
};
unsigned char dq[32] = {
0xA5,0x2F,0xB5,0x50,0xE5,0x43,0x25,0x44,0xC5,0xCA,0xDD,0x59,0x30,0x94,0
x5E,0x88,0x05,0x69,0x09,0xBA,0x88,0x7E,0x22,0xBF,0x26,0x9D,0xB2,0x5C,0x
04,0x4D,0x18,0xB1
};
unsigned char u[40] = {
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xCB,0x55,0x84,0x09,0x27,0x63,0
x7F,0xA9,
0xA9,0x26,0x94,0x23,0x87,0x8B,0x3E,0x36,0x61,0x2D,0xB5,0xCB,0xE2,0x37,0
x6B,0x32,
0xF5,0x84,0xA9,0xC4,0xF0,0xB4,0x1F,0x41
};
unsigned char ciphrtxt[64] = {
0x6E,0xB5,0xC0,0xB0,0x63,0xE8,0x12,0x2B,0x3F,0x17,0xF6,0xAC,0x67,0x23,0
x1E,0xC9,
0x3E,0x64,0x2F,0x9F,0x90,0x45,0xD6,0xF0,0x47,0xD9,0x1D,0x9F,0xCB,0x0C,0
x59,0x90,
0x0E,0x0D,0xA4,0x3D,0x0B,0x44,0x04,0x1A,0x76,0xB4,0x74,0xF0,0x93,0x5E,0
xEE,0x30,
0x70,0x13,0x79,0x24,0xAC,0xE1,0x6A,0x4B,0xA0,0xF3,0x46,0x81,0x89,0x51,0
x81,0x72
};
char output[64] = {0x00};
ica_rsa_modexpo_crt_t icacrt;
unsigned char devName[] = "/dev/z90crypt";
unsigned char extendedName[140] = {0x00};
icacrt.inputdata = ciphrtxt;
icacrt.inputdatalength = sizeof(ciphrtxt);
icacrt.outputdata = output;
icacrt.outputdatalength = sizeof(output);
icacrt.bp_key = dp;
icacrt.bq_key = dq;
icacrt.np_prime = p;
icacrt.nq_prime = q;
icacrt.u_mult_inv = u;
strcpy(extendedName, devName);
dh = open(extendedName,O_RDWR);
if (dh < 0) {
printf("Open of z90crypt failed\n");
}

```

```

return (0);
}
printf("Open of z90crypt succeeded\n");
if ((rv=ioctl(dh, ICARSACRT, &icacrt)) < 0){
printf("rv returned by ioctl: %i\n",rv);
printf("errno given by ioctl: %i\n",errno);
printf("crt ioctl failed\n");
}
else{
printf("rv returned by ioctl: %i\n",rv);
printf ("Output Data Length returned: %i\n",icacrt.outputdatalength);
tell("Output returned: ", output, sizeof(output), 'h');
tell("Raw buffer returned",(unsigned char
*)&icacrt,sizeof(icacrt),'h');
tell("icacrt.outputdata:",icacrt.outputdata,icacrt.outputdatalength,'h'
);
}
if (!(memcmp(icacrt.outputdata+11, msg+11,
icacrt.outputdatalength-11))) printf("Correct result returned.\n");
else
tell("Result incorrect. original msg:",msg+11,
icacrt.outputdatalength,'h');
close(dh);
return (0);}

```

## 7.7.2 icacrtde.c

This program calls libica to perform an RSA decryption using a key token in the CRT format.

```

#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include "tell.h"
#include "ica_api.h"
#define _POSIX_SOURCE 1
#include <unistd.h>
#include <errno.h>
int main ()
/*
static unsigned char expect[] = {
0x00,0x01,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0
xff,0xff,
0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0

```

```

    xff,0xff,
    0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0
    xff,0xff,
    0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0
    xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0
    ff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0x
    f,0x00,0x02,0x3a,0xf3,0x23,
    0x1e,0xc8,0xfe,0xfa,0xe7,0x79,0x14,0x6e,0x1e,0x1e,0xe4,0x26,0x9d,0xad,0
    x3e,0x0a,
    0xfe,0xec,0x48,0x3e,0xa2,0x9f,0x28,0x03,0x89,0x39,0x12,0x82,0xdc,0x39,0
    x5a,0x0e
};

```

```

static unsigned char argu[] = {
0x4c,0xa2,0xcf,0x2f,0x5c,0xb0,0x93,0x80,0x29,0xfe,0x5f,0x0a,0x1b,0x3d,0
x47,0x05,
0xce,0x68,0x5a,0x22,0x44,0xac,0xa0,0x22,0x80,0x2f,0x38,0xb9,0xb0,0x80,0
xf4,0x2a,
0x46,0x4c,0xd1,0x81,0x9e,0xa4,0x1a,0xa3,0xc8,0xf5,0x49,0x43,0xba,0xca,0
xd2,0x59,
0x73,0x18,0xe9,0x72,0xd3,0x24,0x34,0x5b,0xe4,0x26,0x27,0xca,0xd8,0xd0,0
x6e,0xb3,
0x85,0x15,0x95,0x81,0xb1,0xd5,0xfa,0x3a,0x24,0x13,0x59,0x88,0x71,0xa0,0
x1b,0x43,
0x7d,0xee,0xf7,0xce,0xb6,0x01,0xd6,0xc8,0x3c,0x78,0xdf,0x41,0x33,0xbd,0
x6d,0x1d,
0x46,0x51,0x9e,0xfb,0x9a,0x62,0x0a,0x02,0x5d,0xee,0x57,0xd5,0x93,0x8d,0
x74,0xba,
0xa5,0x9c,0xbc,0x34,0x9e,0xcb,0x40,0x5d,0x8c,0xc7,0xc9,0xc0,0xce,0xc4,0
x31,0x8a
};

```

```

static unsigned char argu[] = {
0x65,0x63,0xb2,0x19,0x92,0xec,0xbd,0xf8,0x7e,0xb9,0xa3,0x6d,0x2c,0x8c,0
x53,0x9d,
0x82,0x20,0x5d,0xbe,0x74,0x8d,0xa8,0xe3,0x28,0x07,0xe1,0x15,0xbb,0xe6,0
xbc,0x4f,
0xf5,0xe3,0x98,0xa9,0xd1,0xf8,0x74,0xa6,0xa2,0xbf,0x79,0xc1,0x8b,0xde,0
x90,0x0f,
0x2d,0xa1,0xcf,0x3f,0x4d,0x6f,0xa7,0x03,0xd6,0x76,0xae,0x9f,0x4b,0xde,0
x4c,0xcd,
0x13,0x72,0xd9,0x97,0xcb,0x97,0x33,0x2d,0x04,0x06,0xad,0xe2,0x83,0x4e,0
x74,0x08,
0xde,0xe8,0x04,0x2c,0x9b,0xd2,0xd4,0x08,0x5b,0xab,0xa7,0x18,0x6c,0xc2,0
x4c,0x7e,
0xed,0xe6,0x31,0x12,0xe8,0xcb,0x9b,0x1f,0x1a,0xd1,0xe2,0x52,0xb2,0x9d,0

```



```

x1B,0x8C,
0x2F,0xB2,0xEC,0x6C,0xFE,0xC5,0x60,0x4C,0xFD,0x01,0xF3,0x49,0x49,0xD8,0
x45,0x15,
0x3C,0xD7,0x3B,0x81,0xDC,0xA5,0x08,0x5D,0x9C,0xF8,0xBF,0x70,0x9B,0x69,0
x6A,0x3C
};
unsigned char p[] = {
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xA3,0x49,0xD1,0x8A,0x23,0x0C,0xB0,0x01,0x8B,0xAD,0xFC,0x97,0xD0,0x63,0
xB1,0xD0,
0x82,0x4C,0x9C,0x46,0xA5,0x4B,0x8C,0x8D,0xBB,0x5B,0x9D,0x9A,0xE4,0xFE,0
x2A,0xF3
};
unsigned char q[] = {
0xA5,0x2F,0xB5,0x50,0xE5,0x43,0x25,0x44,0xC5,0xCA,0xDD,0x59,0x30,0x94,
x5E,0x88,
0x05,0x69,0x09,0xBA,0x88,0x7E,0x22,0xBF,0x26,0x9D,0xB2,0x5C,0x04,0x4D,0
x18,0xB1,
}; unsigned char dp[] = {
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xE8,0xBD,0x11,0x73,0xC8,0x0A,0xC6,0x35,0x53,0x77,0x05,0x0C,0x70,0x34,0
xA5,0x70,
0x40,0x59,0x50,0x98,0x90,0xC1,0x82,0x10,0xF7,0xD7,0x27,0x5F,0xE3,0x81,0
x25,0xDF
};
unsigned char dq[] = {
0xE2,0x99,0x6F,0xF1,0xB6,0xEE,0x6A,0x61,0x77,0x4C,0xB3,0xD4,0xCF,0xF0,0
x39,0x46,
0xE8,0x0E,0x05,0xAF,0x10,0x05,0x73,0x54,0xA8,0x0F,0x36,0x6B,0xAC,0x93,0
xD4,0x43
};
unsigned char u[] = {
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xCB,0x55,0x84,0x09,0x27,0x63,0x7F,0xA9,0xA9,0x26,0x94,0x23,0x87,0x8B,0
x3E,0x36,
0x61,0x2D,0xB5,0xCB,0xE2,0x37,0x6B,0x32,0xF5,0x84,0xA9,0xC4,0xF0,0xB4,0
x1F,0x41
};
unsigned char msg[] = {
0x00,0x02,0x8F,0xD7,0x7E,0xC6,0xE0,0xA1,0x24,0xC9,0x48,0xF2,0x29,0xC8,0
x7A,0xEB,
0xDD,0x1E,0xC3,0xE0,0x43,0xE4,0xB3,0x03,0xDB,0x40,0xAC,0x38,0xDA,0x64,0
x3B,0x77,
0x18,0x4D,0xA7,0x5A,0x13,0xFC,0xD0,0x84,0x80,0x50,0x38,0x1F,0xBF,0xFC,0
x56,0x00,

```

```

0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x11,0x22,0x33,0x44,0x55,0x66,0
x77,0x88
};
ICA_ADAPTER_HANDLE dh;           // file handle
unsigned int adapterID = 0;
int rv;                           // return value
char * inpt;                       // input buffer
int inpL;                           // input buffer len
char outp[sizeof(msg)];           // output buffer
unsigned int outL;                 // output buffer length
ICA_KEY_RSA_CRT icacrt;           // input to ica
unsigned char * tgt;               // a temp
icacrt.keyType = RSA_PRIVATE_CHINESE_REMAINDER;
icacrt.keyLength = sizeof(ICA_KEY_RSA_CRT);
icacrt.modulusBitLength = 8 * sizeof(ciphrtxt);
icacrt.pLength = sizeof(p);
icacrt.qLength = sizeof(q);
icacrt.dpLength = sizeof(dp);
icacrt.dqLength = sizeof(dq);
icacrt.qInvLength = sizeof(u);
icacrt.pOffset = (unsigned char *)icacrt.keyRecord -
(unsigned char *)&icacrt;
icacrt.qOffset = icacrt.pOffset + sizeof(p);
icacrt.dpOffset = icacrt.qOffset + sizeof(q);
icacrt.dqOffset = icacrt.dpOffset + sizeof(dp);
icacrt.qInvOffset = icacrt.dqOffset + sizeof(dq);
tgt = (unsigned char *)icacrt.keyRecord;
memcpy(tgt,p,sizeof(p));
tgt += sizeof(p);
memcpy(tgt,q,sizeof(q));
tgt += sizeof(q);
memcpy(tgt,dp,sizeof(dp));
tgt += sizeof(dp);
memcpy(tgt,dq,sizeof(dq));
tgt += sizeof(dq);
memcpy(tgt,u,sizeof(u));
inpt = ciphrtxt;
inpL = sizeof(ciphrtxt);
rv = icaOpenAdapter(adapterID, &dh);
if (dh < 0) {
printf("Open of libica failed\n");
return (0); } printf("Open of libica succeeded\n");
outL = sizeof(outp);
rv = icaRsaCrt(dh,
sizeof(ciphrtxt),

```

```

    ciphrtxt,
    &icacrt,
    &outL,
    outp);
    if (rv < 0){
    printf("rv returned by libica: %i\n",rv);
    printf("ioctl failed\n");
    }
    else{
    printf("rv returned by libica: %i\n",rv);
    printf ("Output Data Length returned: %i\n",outL);
    tell("Output returned: ", outp, sizeof(outp), 'h');
    }
    if (!(memcmp(msg, outp, sizeof(msg))))
    printf("Correct result returned\n\n");
    else {
    tell("Incorrect result--expected output:", msg, sizeof(msg), 'h');
    tell("Output returned: ", outp, sizeof(outp), 'h');
    }
    icaCloseAdapter(dh);
    return (0);
    }

```

### 7.7.3 tell.h

```

#ifndef _TELLH_
#define _TELLH_ 1
#include <string.h>
int printcl(unsigned char *, unsigned int);
int printrw(unsigned char *, unsigned int);
int printhx(unsigned char *, unsigned char *, unsigned int);
int tell(unsigned char *, unsigned char *, unsigned int, unsigned char);

#endif

```

### 7.7.4 tellit.c

```

#include "tell.h"
#include <stdlib.h>
#include <stdio.h>
int tell(unsigned char * title,
unsigned char * data,
unsigned int dl,
unsigned char fmt)

```

```

/*****/
/*
/* tell: writes input title and data to stdout */
/*
/* Input: title_str -- character string to be written before data */
/*
/* data -- data to be written */
/* note that data will *always* be converted */
/* to ascii hex characters. */
/*
/* dl -- length of data to be written */
/*
/* fmt -- format: 'd' means decimal */
/* 'h' means hex */
/*
/* Output: screen is updated */
/*
/* Process: 1. skip a line */
/* 2. write title */
/* 4. if dl is non-zero */
/* convert data to ascii hex and */
/* write data in 4 8-byte (16 chr) chunks per line*/
/* (indent 4 and skip 2 between chunks, leaving */
/* a right margin of 6) */
/*
/*****/
{
printhx(title, data, dl);
return 1;
} /* end tell */
print printhx(unsigned char * title,
unsigned char * addr,
unsigned int len)
{
int hl = 0;
int inl = 0;
int r,rx;
printf("\n%s\n",title);
hl += strlen(title)+2;
for (r=0; r<len/32; r++) {
hl += printrw(addr+inl, 32);
inl += 32;
}
rx = len%32;
if (rx) {

```

```

hl += printrw(addr+inl, rx);
inl += rx;
}
printf("\n");
hl++;
return hl;
}
int printrw(unsigned char * addr,
unsigned int len)
{
int hl = 0;
int inl = 0;
int c, cx;
printf("  "); // left margin
hl += 4;
for (c=0;c<len/8;c++) {
hl += printcl(addr+inl, 8);
inl += 8;
}
cx = len%8;
if (cx) {
hl += printcl(addr+inl, cx);
inl += cx;
}
printf("\n");
hl++;
return hl;
}
int printcl(unsigned char * addr,
unsigned int len)
int hl = 0;
int i;
for (i=0;i<len;i++) {
printf("%02x", (unsigned int) addr[i]);
hl += 2;
}
printf(" ");
hl++;
return hl;
}
int printcll(unsigned char * addr,
unsigned int len)
{
int hl = 0;
int i;

```

```

for (i=0;i<len;i++) {
printf("%01x",(unsigned int) addr[i]);
hl += 1;
}
printf(" ");
hl++;
return hl;
} // end printc1l
int printrw1(unsigned char * addr,
unsigned int len)
{
int hl = 0;
int inl = 0;
int c, cx;
printf("    "); // left margin
hl += 4;
for (c=0;c<len/16;c++) {
hl += printc1l(addr+inl, 16);
inl += 16;
}
cx = len%16;
if (cx) {
hl += printc1l(addr+inl, cx);
inl += cx;
}
printf("\n");
hl++;
return hl;
} // end printrw1
int printhx1(unsigned char * title,
unsigned char * addr,
unsigned int len)
{
int hl = 0;
int inl = 0;
int r,rx;
printf("\n%s\n",title);
hl += strlen(title)+2;
for (r=0; r<len/64; r++) {
hl += printrw1(addr+inl, 64);
inl += 64;
}
rx = len%64;
if (rx) {
hl += printrw1(addr+inl, rx);
}
}

```

```

inl += rx;
}
printf("\n");
hl++;
return hl;
} // end printhx1
int tell1(unsigned char * title,
unsigned char * data,
unsigned int dl,
unsigned char fmt)
/*****
/*
/*      tell: writes input title and data to stdout      */
/*
/* Input:  title_str -- character string to be written before data */
/*
/*          data -- data to be written                    */
/*          note that data will *always* be converted    */
/*          to ascii hex characters.                    */
/*
/*          dl -- length of data to be written           */
/*
/*          fmt -- format: 'd' means decimal             */
/*                  'h' means hex                       */
/*
/* Output:  screen is updated                            */
/*
/* Process:  1. skip a line                              */
/*           2. write title                              */
/*           4. if dl is non-zero                        */
/*              convert data to ascii hex and           */
/*              write data in 4 8-byte (16 chr) chunks per line*/
/*              (indent 4 and skip 2 between chunks, leaving */
/*              a right margin of 6)                    */
/*
/*
/*****
{
printhx1(title, data, dl);
return 1;
} /* end tell */

```

### 7.7.5 makecrtde

```
# This is the makefile for the tstcrtde
INCS = -I.
CFLAGS = -DZ90CRYPT_DEBUG -D_REENTRANT $(INCS) -O -g -Wall
OBJS = tstcrtde.o tellit.o
tstcrtde: $(OBJS)
gcc -o tstcrtde $(OBJS)
clean:
rm -f tstcrtde.o tstcrtde
```

### 7.7.6 makeicacr

```
# This is the makefile for the icacrtde
CFLAGS = -DZ90CRYPT_DEBUG -D_REENTRANT -O -g -Wall
OBJS = icacrtde.o tellit.o
icacrtde: $(OBJS)
gcc -licr -o icacrtde $(OBJS)
clean:
rm -f icacrtde.o icacrtde
```





## PCICC User Defined Extensions (UDX)

UDX is the facility offered by the S/390 and zSeries PCICC, when running under control of ICSF, that allows users to design and implement their own cryptographic service(s) to be executed in the PCICC itself. This provides maximum flexibility to the PCICC user, along with all the protection that the card can offer.

## UDX overview

A UDX implementation on zSeries requires three parts:

- ▶ An ICSF callable service, which when called by a user application will invoke the UDX code being executed in the PCICC. This is a user defined host service that matches the specific UDX code running in the card. Applications call the service using a designated number.
- ▶ The PCICC UDX code, which is the coprocessor piece installed by the user into the PCICC code itself, and is intended to be invoked by the ICSF service. A UDX is designated by a specific two-character identity.
- ▶ A service “stub”, to be link-edited with the application calling the user defined service for proper dispatching of the service by ICSF.

Additionally, the specific UDX service can be enabled or disabled using access control points managed from the TKE Workstation. This requires that you also design and install an exit to the CSFPCI (TKE communication) ICSF-callable service.

The Transaction Security System IBM 4753 Network Security Processor (NSP), now withdrawn from marketing, supported the User defined Function (UDF) and User Developed Program (UDP). The PCICC, designated as the functional follow-on to the NSP, supports UDX. It does not support UDF and UDP.

## PCICC code structure and UDX

As shown in Figure A-1 on page 297, the PCICC code is organized by “segments” in the card flash memory. Segment 3 contains the code executing the set of cryptographic functions provided by the API the customer wishes to use.

For zSeries, the set of functions and API is imposed, by IBM, to be the IBM CCA services and is implemented as a set of “CCA command processors” in segment 3. Note that other versions of the 4758 cryptographic adapter, running on other platforms, give the user the choice to load a Segment 3 code offering either the CCA or PKCS#11 API.

The UDX code to be installed in a PCICC consists of a user-designed command processor that is link-edited into the Segment 3 load module during the UDX generation process. Therefore, creating a UDX requires you to get the proper tools for designing, debugging, and packaging a new Segment 3 load. As the base Segment 3 load, a UDX load is checked for origin and integrity using a digital signature scheme. The new Segment 3 load, with the UDX integrated, must have been signed by an IBM-approved key.

Note that a customer may elect to have more than one UDX in a single coprocessor. Each UDX will be uniquely referred to using its function identifier (refer to “ICSF and PCICC communications” on page 298).

Segment 2 contains the CP/Q++ operating system which interfaces with ICSF for getting the function request and issuing the related response. The CCA Manager receives requests that come from the host, decides what they are, and dispatches the appropriate command processor. CP/Q++ is the operating system/device drivers, that allows application programs (like CCA) to make requests to the on-card hardware including cryptographic hardware, communications hardware (over the PCI bus), Flash memory, etc.

The CCA Manager component of the CCA application program in the card uses an internal table containing all the entry points for the CCA and installed UDX command processors. This table is updated following a coprocessor reset sequence. Part of the control done prior to dispatching a command processor is the permission given to the 4758 role to use the command processor; in the case of the PCICC, the role is always the DEFAULT role.

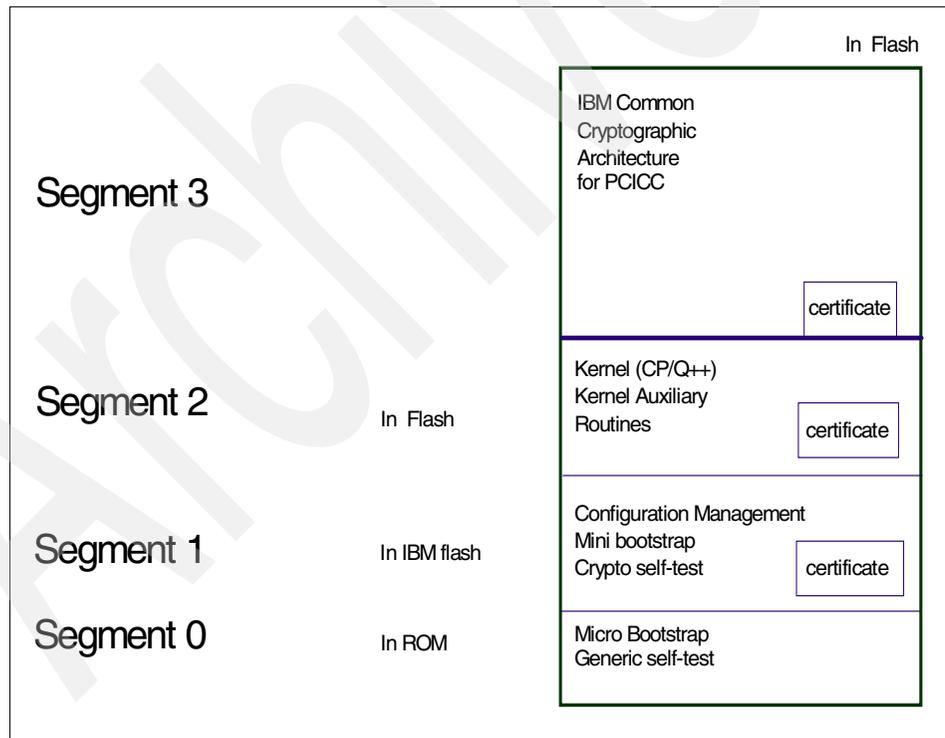


Figure A-1 PCICC internal code structure

## ICSF and PCICC communications

ICSF is initially called by a cryptographic application using a CCA “verb” (another name for the callable services). To process the requested service, ICSF issues specific instructions that are non-disclosed S/390 instructions, which result in sending a request to the selected coprocessor. If the selected coprocessor is a PCICC, the request is sent as a Cooperative Processing Request/Response Block (CPRB) that is interpreted by CCA in the PCICC.

A CPRB contains a sub-function code, a two-character value, which is used to uniquely identify the required command processor. After parsing the received CPRB, CCA uses the sub-function identifier to invoke the proper command processor. This is shown in Figure A-2. After providing the service, the PCICC sends the response back, still as a CPRB.

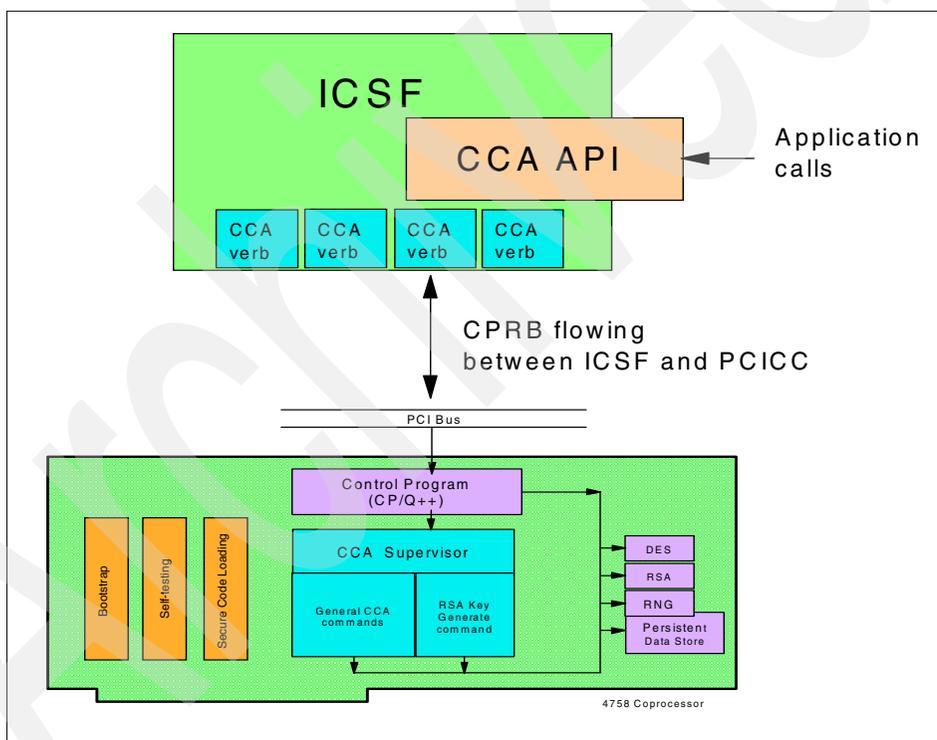


Figure A-2 PCICC internal operating environment

## UDX invocation

In the following sections, we explain UDX invocation.

## UDX function code identifier

The UDX is in essence a command processor, like the CCA command processors, which has its own function identifier. (The following range of identifiers is reserved for UDX: XA-XZ, X0-X9, YA-YZ, Y0-Y9, WA-WZ, W0-W9.)

The function identifier corresponding to an installed UDX is known to CCA following a coprocessor reset; however, the ICSF code shipped with z/OS cannot integrate in advance the function code that users may choose for their UDX.

This is solved by adding the following UDX statement to the Options Data Set (once the UDX has been installed in the PCICC):

```
UDX(UDX-id,service-number,load-module-name,'comment_text',FAIL(failoption))
```

Where *UDX-id* is the function code that identifies the UDX command processor according to the function code specified during the UDX generation process; *service-number* is to be used internally to refer to this callable service (see the stub in the following section “The UDX callable service and the stub” on page 299 for more details); *load-module-name* is the name of the user-defined callable service that calls the UDX in the PCICC (refer to the *ICSF System Programmer's Guide* for a complete description of the statement).

## The UDX callable service and the stub

A UDX, not being a base CCA “verb”, requires a specific routine in ICSF to provide the service at the CCA API level. This ICSF callable service in turn manages to call the UDX in the PCICC.

The capability of creating customized services in ICSF (the “Installation-defined services”) has been available since the early releases of ICSF. These services are customer-written modules link-edited with ICSF and identified in the Options Data Set (refer to the *ICSF System Programmer's Guide*, SA22-7520) by the SERVICE statement where a unique number (from 1 to 32767, inclusive) is associated to the service load module. (Refer to the *ICSF System Programmer's Guide*, SA22-7520, for more information about the Options Data Set.)

After you write the callable service, you need to link-edit it into a load module and install the load module into an APF-authorized library.

ICSF uses the following normal search order to locate the service:

- ▶ Job pack area
- ▶ Steplib (if one exists)
- ▶ Link pack area (LPA)

- ▶ Link list (SYS1.LINKLIB concatenation)

During ICSF startup, ICSF loads the load module that contains the service into the ICSF address space with the ICSF callable services. ICSF binds the service with the service number that you specified in the installation options data set.

To call such an installation-defined service, the application has to call an intermediary piece of code called the “stub”. The service stub is designed and installed by the user, and it must do the following:

- ▶ Check that ICSF is active.
- ▶ Place the service number for the installation-defined callable service into register 0.
- ▶ Call the IBM-supplied processing routine, CSFAPRPC, which is internally used by ICSF to access the callable services. The stub must thus first retrieve the location of CSFAPRPC from the ICSF cryptographic communication vector table (CCVT).

A stub example is provided in the *ICSF System Programmer's Guide*.

Any application program that calls a service stub must be link-edited with the service stub. To call an installation-defined service from an application program, use the following statement:

```
CALL <service-stub-name><service-parameters>
```

The service-stub-name is the name of the service stub for the installation-defined callable service. The service-parameters are the parameters you want to pass to the installation-defined service. You supply the parameters according to the syntax of the programming language that you use to write the application program.

Starting with OS/390 R10, this capability is extended to allow you to interact with the UDX in the PCICC by installing a user-defined ICSF module that will request the PCICC to run the UDX in Segment 3; this is illustrated in Figure A-3.

This “UDX Callable Service” must also have a unique number that is not shared with another UDX or non-UDX installation-defined service; the service number is specified in the UDX statement of the Options Data Set, as indicated in “UDX function code identifier” on page 299. As for a regular installation-defined service, the service invocation is performed via the stub.

Note that with a UDX callable service, the actual service is performed by the UDX code running in the card—whereas with a regular user-defined service, the service is provided by the code running in the ICSF address space.

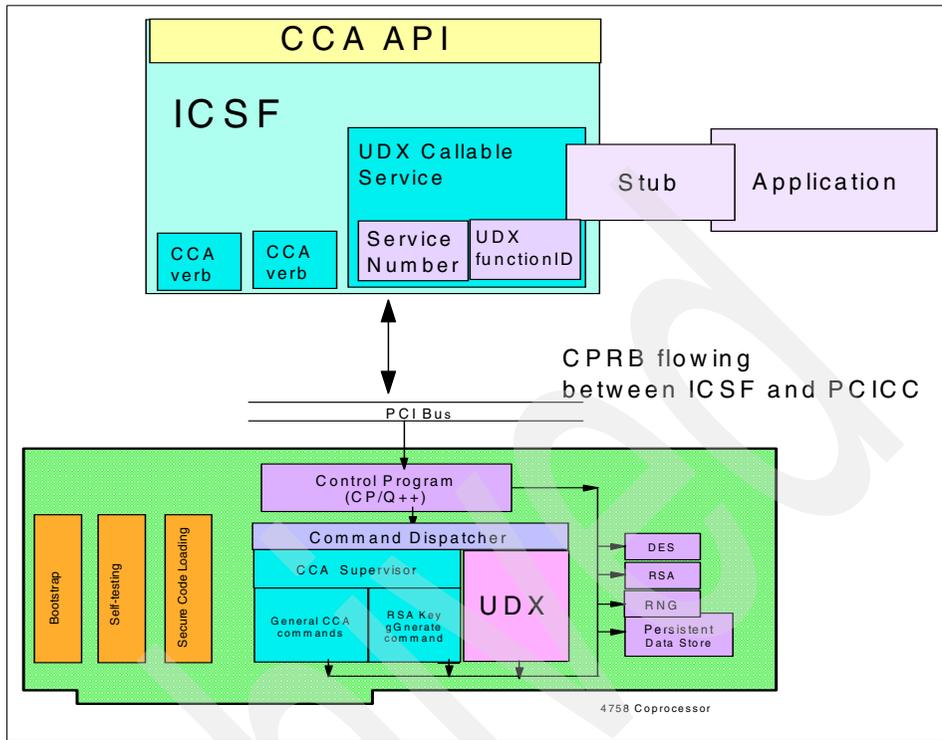


Figure A-3 PCICC internal operating environment and UDX

# The UDX development process

## What the UDX does, and how

A PCICC UDX is a piece of code to be run by the PCICC. It performs the logic of a cryptographic process to be run against the data received from ICSF in the CPRB. The actual cryptographic functions executed during this process can be coded in the UDX code itself, or the UDX code can invoke the native PCICC cryptographic and storage functions via a lower-level API.

The PCICC UDX source is written in C, and a specific IBM Toolkit is needed that provides the required compilation libraries, debugging and packaging tools. Typically a UDX is developed and tested on a workstation with a 4758 card and, when completed and packaged, eventually loaded into the user zSeries system.

## The PCICC UDX development process

With the zSeries systems and z/OS V1R2, the UDX development process reaches its “phase 2”.

In the “phase 1” process, available with 9672 G5/G6 enterprise servers and OS/390 R10, IBM Global Services (IGS) designs the UDX in cooperation with the customer, and the IBM S/390 Laboratory tests and packages the PCICC UDX code as an S/390 Licensed Internal Code (LIC) update. The update is distributed to all 9672 systems in the field via the Microcode Change Level (MCL) process, and users are given a password (that is, a key) to load and enable “their” UDX into the PCICC.

To initiate the process, the customer makes an RPQ request to get a PCICC UDX. During the generation process, the PCICC UDX code is signed by IBM and the signature is verified when loading the code into the PCICC. (Note that in this context, IBM owns the UDX.)

The ICSF UDX callable service, the service stub, and the optional access control point exit with the PCICC code can be designed with IBM support. This redbook addresses UDX phase 2 support only.

**Note:** zSeries systems can receive UDX developed and distributed (via LIC change) by IBM. In that case, the customer will have to enter the password provided by IBM in the User Defined Extension Management ISPF panel to authorize use of the UDX on a given PCICC.

In contrast, what we describe in this book is the user-developed UDX approach, where the UDX is already authorized to the PCICC it has been loaded into.

## UDX process phase 2 support

In the following sections, we describe the UDX process phase 2 support in more detail.

### PCICC UDX generation process overview

UDX phase 2 support provides the ability for customers to develop their own UDXs, with the implication that they own their UDX but are also able to sign it with a key that is acceptable for loading into the PCICC. This support also requires the PCICC to be installed in a zSeries system running z/OS R2.

After submitting an RPQ to IBM, the customer is provided with a UDX Development Toolkit and signing keys. The key phases of the PCICC piece of the UDX are summarized as follows:

1. Define the UDX command processor API.
2. Define access control points for the UDX.
3. Define new completion codes for the UDX.
4. Define the subfunction code for the UDX.
5. Add the UDX command processor to the command decoding array.
6. Design and code the logic of the coprocessor piece of the UDX.
7. Build the UDX coprocessor executable, as shown in Figure A-4 on page 304.

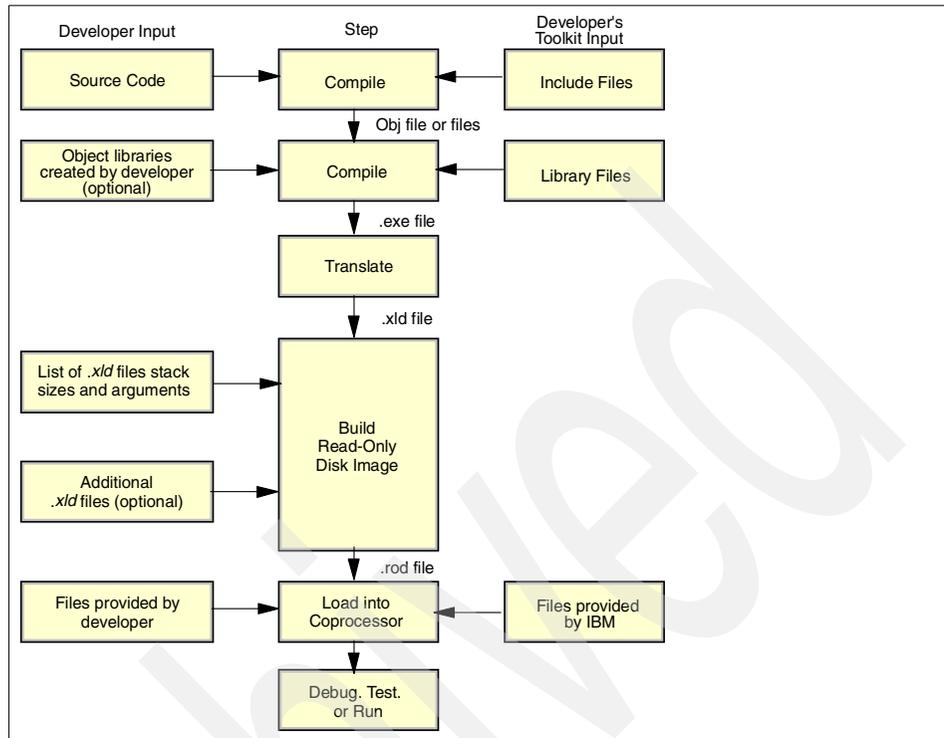


Figure A-4 The PCICC UDX development process

## Building the UDX coprocessor executable

The IBM Toolkit is available to the customer under contract. It runs on a workstation with Windows NT or Windows 2000 and a 4758 plugged in. Its main components are:

- ▶ A translator, to transform the executable files into a format (.xld) required by CP/Q++
- ▶ An image builder, which packages executables into a read-only disk image
- ▶ A debugger (to test and control, on an NT workstation, the code that is to eventually execute into the PCICC) and a file browser
- ▶ A special development code loader, to load the packaged code in a coprocessor configured as a development platform on the workstation
- ▶ A set of libraries and header files, to be used during compilation and link-edit
- ▶ A signer program, to enable the signing of the code load by an IBM-approved key

**Note:** There is no compiler or binder in the Toolkit, because the customer is to use IBM VisualAge® C++ or Microsoft Visual C++ for performing these steps.

**Important:** The output of the PCICC UDX generation process is a complete Segment 3 load (that is, UDX *and* the regular CCA command processors, all link edited into a single load).

As a consequence, each PCICC installed in a zSeries system may have a unique Segment 3 code load. Segment 3 may be the IBM image or a custom image. A single customer may create multiple custom Segment 3 images.

### ***Approval of the customer code signing key***

After generating their RSA key pair, customers send the public key to IBM. In turn, IBM sends the certificate for the key. The key is certified by an IBM authority in the verification chain imbedded in the PCICC code, and the received certificate will be in the Segment 3 load.

## **Installing the PCICC UDX**

Following is a brief summary of the PCICC UDX installation process:

1. The PCICC targets for UDX loading are deactivated at the ICSF panel.
2. The UDX file on diskette (the custom Segment 3 load) is imported via the HMC or the system Support Element.
3. The UDX code load is activated (that is, loaded) into one or more PCICCs.
4. The target PCICCs are activated again at the ICSF panel.

This is concurrent with system installation. At any time, the customer may elect to reload the original IBM Segment 3 using the “Reset UDX to IBM image” function at the HMC or SE.

**Note:** If a crypto problem is experienced, customers may be required to remove their UDX and recreate the problem before IBM will troubleshoot.

## **Designing and developing the host piece of the UDX**

In the following sections we describe in more detail what you need to do in order to design and develop the host piece of the UDX.

## The ICSF callable service and the service stub

The host piece of a UDX is typically straightforward; it essentially constructs a request block, sends the block to the coprocessor, and parses the result. Now let's look at the development more closely.

### Development

In general, the host piece of a UDX should be as small as possible. Most of the work should be performed by the coprocessor piece.

The UDX callable service module, when called by an application program via the stub, typically performs the following tasks:

- ▶ Checks its input parameters.
- ▶ Constructs a request block.
- ▶ Sends the request to the coprocessor and receives the reply.
- ▶ Extracts the result, and returns the result to the user's application.

During ICSF startup, ICSF loads the load module containing the UDX service into the ICSF address space with the ICSF callable services. ICSF binds the service with the service number specified in the Installation Options Data Set.

To create the host piece of the UDX, follow these steps:

1. Define the UDX API.
2. Define the subfunction code for the UDX. There will be one subfunction code associated with the UDX command processor. The following 2-character code points have been reserved for CCA extensions (you must not use other codepoints, as they may conflict with existing CCA commands):

WA - WZ, W0 - W9

XA - XZ, X0 - X9 (reserved for customer-written UDXs)

YA - YZ, Y0 - Y9 (reserved for customer-written UDXs)

3. Define new completion codes for the UDX.
4. Update the appropriate ICSF macros supplied with the UDX Development Toolkit for zSeries with the UDX subfunction code and completion codes.
5. Design and code the logic of the UDX callable service.
6. Code the UDX service stub.
7. Compile the UDX callable service and service stub.

Note that a set of ICSF macros is delivered with the UDX Toolkit to provide the UDX callable service with ICSF standardized communication functions; these are:

- ▶ CSFACKDS - Access the in-storage ICSF Cryptographic Keys Data Set.

- ▶ CSFAPKDS - Access the ICSF Public Key Data Set.
- ▶ CSFACCPN - Send a request to the coprocessor.
- ▶ CSFACPRB - Build a CPRB.
- ▶ CSFADSCP - Destroy a CPRB.
- ▶ CSFAVLPB - Validate a CPRB.
- ▶ CSFAPBLK - Parse a CPRB.
- ▶ CSFAPKTV - Validate/initialize an RSA/DSS key token.
- ▶ CSFADSPI - Communication interface between services and the coprocessor.
- ▶ CSFASEC - Check authorization to a RACF-protected or security exit-protected resource.

## Installation

1. The OBJ file for the UDX callable service must be link-edited into a load module and installed into an APF-authorized library. ICSF uses the normal OS/390 search order to locate the service:
  - Job pack area
  - Steplib (if one exists)
  - Link pack area (LPA)
  - Link list (SYS1.LINKLIB concatenation)
2. The OBJ file for the service stub must be link-edited with the application program that calls the service stub. Any application program that calls a service stub must be link-edited with the service stub. To call a UDX service from an application program, use the following statement:
 

```
CALL <service_stub_name> <service_parameters>
```

where `service_stub_name` is the name of the service stub for the UDX callable service, and `service_parameters` are the parameters you want to pass to the UDX callable service. You supply the parameters according to the syntax of the programming language that you use to write the application program.
3. You must identify the UDX service in the ICSF Installation Options Data Set using the UDX keyword. For information about the specification of the UDX keyword, refer to *z/OS ICSF System Programmer's Guide, SA22-7520*.
 

You will specify information including the UDX subfunction code, a service number, and a load module name. Figure A-4 on page 304 shows an excerpt of the Options Data Set that we used during our testing.

```
UDX(XX,50,UDXSRV,'TEST UDX FOR CRYPTO RESIDENCY',FAIL(YES))
EXIT(CSFPCI,UDXEXIT,FAIL(NONE))
```

Figure A-5 UDX service and exit in Options Data Set

## The access control point exit

Access control points are switches associated to the command processors, for the DEFAULT profile, in the PCICC that allow or disallow access to the service provided by the command processor. These switches can be turned on or off from the TKE Workstations.

The list of access control points, and their individual status, for the regular CCA command processors is provided as the response to a “Query Access Control Points” request from ICSF. They can then be displayed with informational data. Actually, ICSF receives from the PCICC the hex value for the access control point and uses internal tables to find the text description of this specific access control point.

The UDX access control point information cannot be installed into the ICSF internal tables. It takes a user-installed postprocessing exit on the CSFPCI service (the PCI interface) to establish the mapping between the UDX access control point hex value and its clear text information.

The exit checks if the function just executed was a “Query Access Control Points”. If it is, the exit appends to the PCICC response its own piece of the description table.

Figure A-6 on page 309 shows an excerpt of a CSFPCI exit example, where the mapping between the hex value (A01\_CODE) of the access control point and its ASCII (to be displayed on the TKE Workstation) description (A01\_TXT) is done.

```

*-----
          ORG  UACPTS+9          A01 STARTING OFFSET
A01      DS   CL38              A01 LENGTH
          ORG  A01              A01 BREAKDOWN
A01_TYPE DC  X'02'             A01 TYPE -->MUST REMAIN X'02'
A01_CODE DC  X'8888'           A01 ACCESS CONTROL POINT
A01_TXT_LEN DC X'00000008'     A01 TEXT LENGTH
A01_TXT   DC  X'4954534F20554458'
*
A01_FLAG DC  X'00000000'       A01 ASCII DESCRIPTION OF ACP
A01_FLAG DC  X'00000000'       A01 FLAGS
A01_ACPTS_CNT DC X'00000000'   A01 ACPS ENABLE COUNT - NONE
A01_ACPTS DS  OF               A01 ACPS NEEDING ENABLED - NONE
*-----

```

Figure A-6 UDX access control point description in the exit

When displaying the corresponding PCICC domain controls on the TKE, you will see the panel shown in Figure A-7.

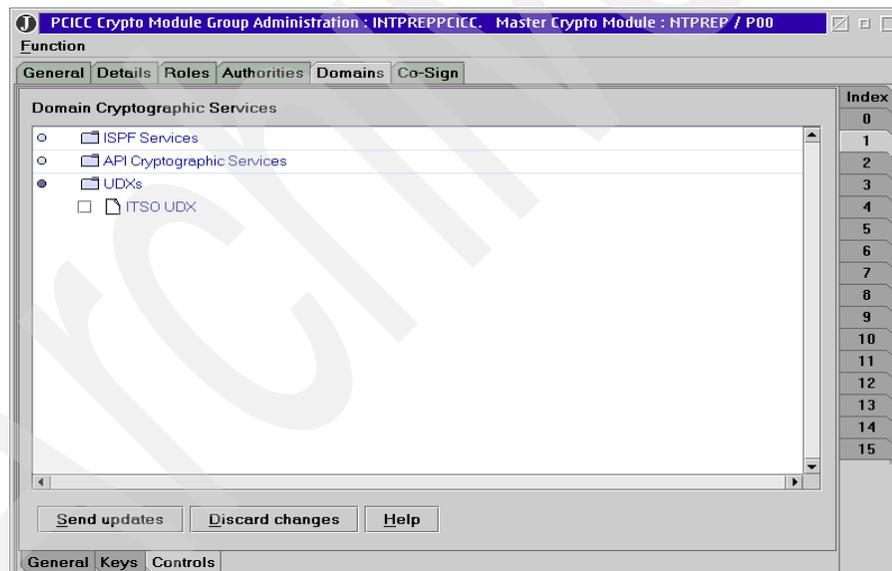


Figure A-7 Displaying the UDX access control point at the TKE

The exit must be installed by link-editing the OBJ file into a load module and installing the load module into an APF-authorized library.

ICSF uses the normal z/OS search order to locate the service:

- Job pack area
- Steplib (if one exists)
- Link pack area (LPA)
- Link list (SYS1.LINKLIB concatenation)

The ICSF Installation Options Data Set must be updated to define the exit. Use the EXIT keyword, specifying CSFPCI for the ICSF name of the callable service exit.

For information about the specification of the EXIT keyword, refer to *z/OS ICSF System Programmer's Guide* and also see the example in Figure A-5 on page 308.

# Callable services access control points

## The access control points

The access to CCA services provided by the PCICC can be controlled by access control points since OS/390 V2R9, with APAR OW46381, TKE code V3.0 with ECA 186 (EC H25130), and zSeries LIC driver 36J EC# H25104 MCL# P104F002 and P104F003.

The access control points are defined in the PCICC DEFAULT role, the role under which the PCICC executes the ICSF requests for each domain. They can be thought as switches allowing or disallowing access to the PCICC command processors providing the requested CCA service, and working inside the PCICC hardware—as opposed to software-controlled access using the CSFSERV class of profiles in RACF or equivalent.

Switching an access control point between the enabled and disabled state can only be done using a TKE Workstation, and in the domain controls panel, where all the recognized access control points are listed with the related CCA service name.

**Note:** A non-TKE installation will not “see” the access control points that will all appear as being enabled to the PCICCs—except for the access control point of the DKG-DALL service (Diversified Key Generate for all key types), which comes disabled and requires a TKE to enable access to the service.

Clicking the individual check box flips the access control point between the enabled or disabled state in the TKE display. The new state is actually recorded into the PCICC domain with the “send updates” button, provided that the authority is permitted to “domain controls” and proper “domain access” in its role.

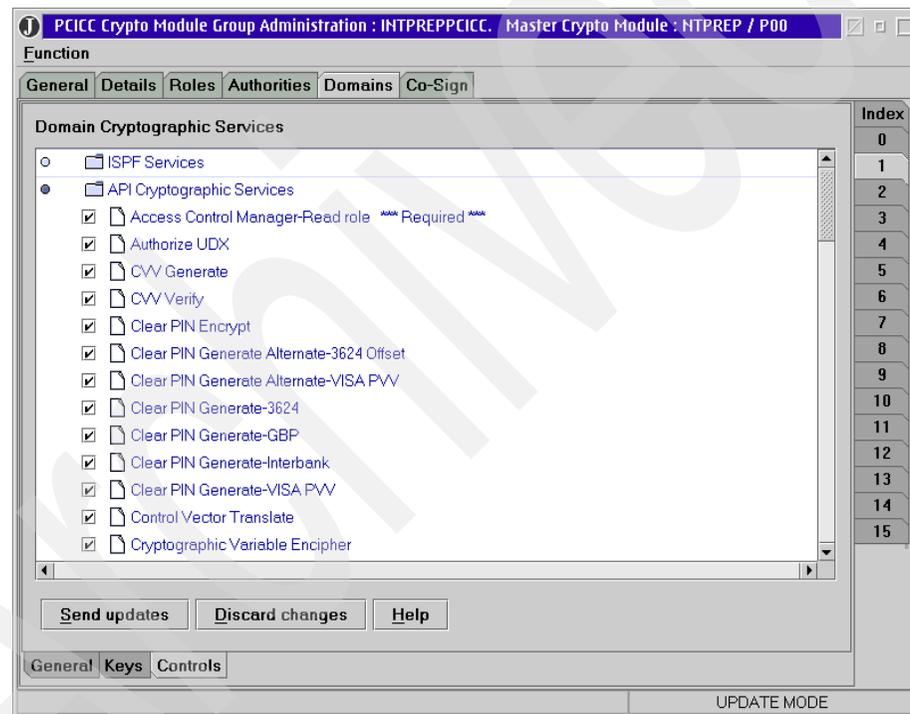


Figure B-1 Callable Services access control points in the domain controls

Figure B-2 on page 313 shows the access control point for a Symmetric Key Generate PKCS-1.2 in the disabled state.

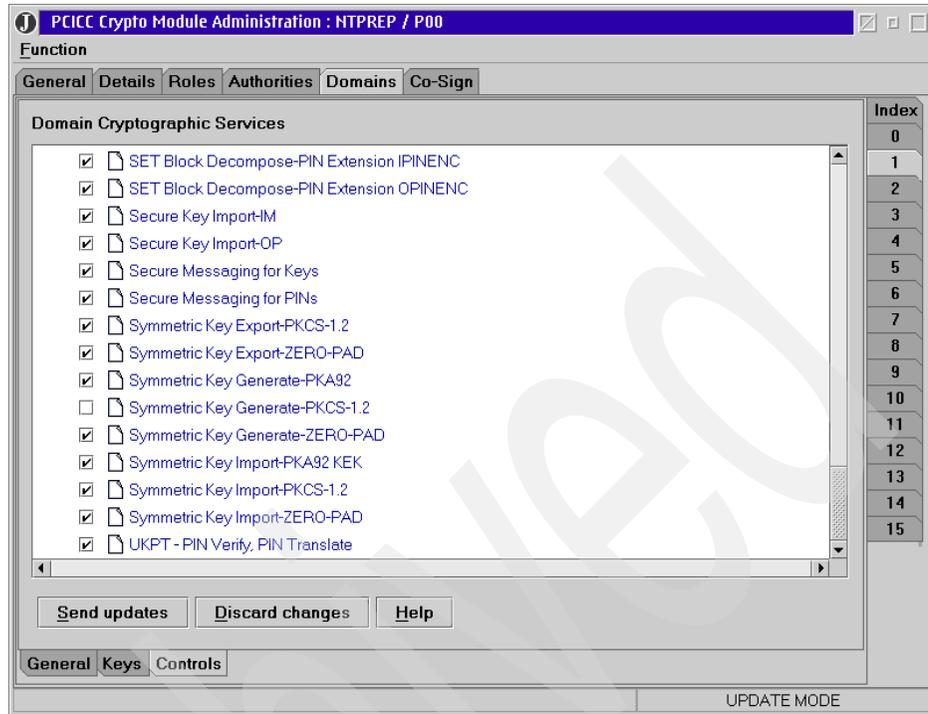


Figure B-2 Access to Symmetric Key Generate disabled

A short test program performing the Symmetric Key Generate ends with an error code, as shown in Figure B-3.

```

Normal execution of the test program
+SYMMETRIC KEY GENERATE - GENERATING
+KEY GENERATION DONE
+DES TOKEN WRITTEN
+RSA ENCIPHERED KEY WRITTEN

After disabling the access control point:

+SYMMETRIC KEY GENERATE - GENERATING
+ERROR CODE = 00008, REASON CODE = 00090

```

Figure B-3 Test of a disabled access control point

**Important:** We found out the hard way that, even with REASONCODES(ICSF) specified in the Options Data Set, the denied by access control point reason code must be searched for in the TSS reason codes (refer to Appendix A in the *ICSF Application Programmer's Guide*, SA22-7522).

You will notice that some access control points are indicated as **\*\*\*required\*\*\*** in the TKE display. If you attempt to modify them, the pop-up window shown in Figure B-4 will appear.

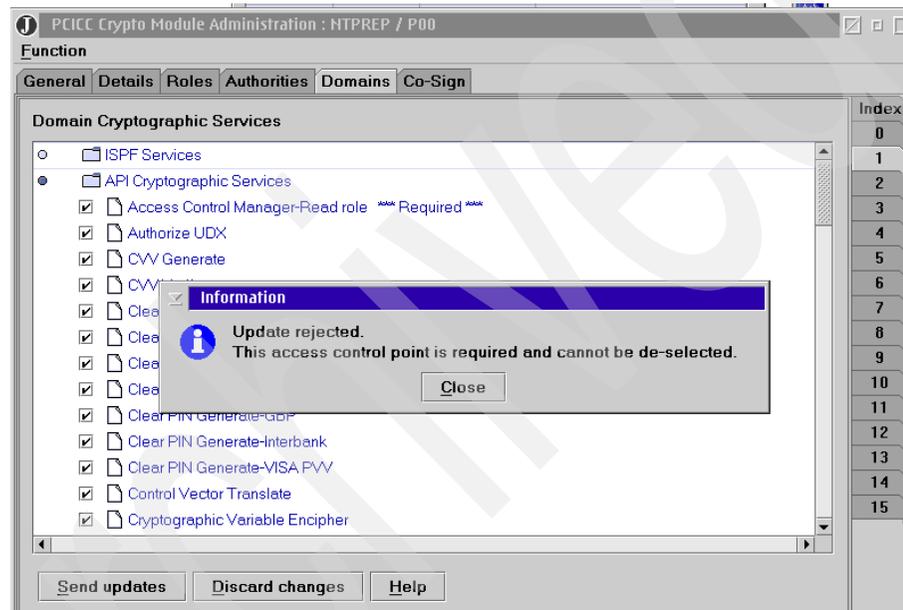


Figure B-4 Required access control points

The access control points can also be displayed at the ICSF ISPF panels by marking a PCICC in the Coprocessor Management panel with an R (display default role), as shown in Figure B-5 on page 315.

The next panel, shown in Figure B-6 on page 315, displays all the access control points which are currently enabled in the DEFAULT role of the selected coprocessor for the pertinent domain.

```

----- ICSF Coprocessor Management ----- Row 1 to 6

Select the coprocessors to be processed and press ENTER.
Action characters are: A, D, E, R, and S. See the help panel for details.

      COPROCESSOR      MODULE ID/SERIAL NUMBER      STATUS
      -----
.   A02                                     ACTIVE
.   A03                                     ACTIVE
.   C0          04100000000039C4 04100000000039C4  ACTIVE
.   C1          0410000000003992 0410000000003992  ACTIVE
r   P00          92E01846                                     ACTIVE
.   P01          92E01983                                     ACTIVE

```

Figure B-5 Selecting the default role display in a PCICC

```

----- ICSF - Status Display ----- Row 69 to
COMMAND ==>

Enabled access control points from the default role for P00 domain 1

Secure Messaging for PINs
Set ASYM Master Key
Set SYM Master Key
Symmetric Key Export - PKCS-1.2
Symmetric Key Export - ZERO-PAD
Symmetric Key Generate - PKA92
Symmetric Key Generate - PKCS-1.2
Symmetric Key Generate - ZERO-PAD
Symmetric Key Import - PKA92 KEK
Symmetric Key Import - PKCS-1.2
Symmetric Key Import - ZERO-PAD
SET Block Compose
SET Block Decompose
SET Block Decompose - PIN Extension IPINENC
SET Block Decompose - PIN Extension OPINENC
TKE Authorization for domain 1

```

Figure B-6 ISPF panel showing enabled access control points

Access control points can also be defined for UDX services, as described in “Access control points in the PCICC and ICSF” on page 316.

## Access control points in the PCICC and ICSF

Access control points are known to the PCICC and ICSF by a 4-digit hexadecimal number. There is actually a table in ICSF that maps the access control points values to a text description. This table is used both to initialize new access control points in the PCICC, and to externalize the name of the access control point.

## New access control points and TKE users

Prior to TKE 3.1, there was no mechanism available to enable/disable the access control points for services that were executed on the PCICC. Therefore, the default role had all the access control points enabled. With TKE 3.1, IBM provides the customer with the ability to disable/enable the access control points.

The PCICC code records the fact that it has been accessed once by a TKE and that will influence the state of the access control points as seen by new TKE users or brought by new services, as described in the following sections.

### Non TKE users

Non-TKE users have all the access control points enabled in the DEFAULT role, except for the DKG-DALL service and any UDX access control points (these services do require a TKE to be enabled).

### New TKE users

For new TKE users, (that is, users who never installed a TKE before), their PCICCs never recorded that they were accessed from a TKE and the access control points are initially set as if they were non-TKE users (that is, all enabled except for the DKG-DALL and UDX access control points).

### TKE users

For TKE users, the PCICCs have recorded that they have been accessed from a TKE once and, when new access control points appear in ICSF, the DEFAULT profile will not be refreshed for the new access control points. As a consequence, access control points for the already existing services will be set to enable (still with the exception of DKG-DALL and UDX), and access control points for new services will be disabled.



## Exploitation of the cryptographic coprocessors

In this appendix, we provide brief descriptions of the IBM software products that exploit zSeries hardware cryptography.

## Exploitation of the zSeries CCFs and PCI coprocessors

The main API to exploit the zSeries hardware cryptographic coprocessors is ICSF. ICSF dynamically routes the requests to either the CCF, the PCICC or the PCICA transparently to the application.

Application designers cannot specifically decide to use one of these coprocessor types, unless the application requests services or key formats only available with the PCICC (as described in the *ICSF Application Programmer's Guide*), or invokes the CSNDPKD service (the SSL handshake assist) using a clear private RSA key and with PCICA card(s) active in the system.

The z/OS Base Cryptographic Services provide access to the hardware cryptographic coprocessors in the following three ways, as shown in Figure C-1 on page 319.

- ▶ ICSF is the most direct path for an application to get access to the coprocessors, by calling the ICSF IBM CCA services. ICSF requires at least one CCF to be active in the system; otherwise, it will not start.
- ▶ Open Cryptographic Service Facilities (OCSF) is a z/OS implementation of the Intel CDSA (Common Data Security Architecture) API. The intent of the CDSA API is mostly to provide the services required in a PKI (Public Key Infrastructure) environment. These services are actually given by OCSF service providers' plug-in modules.

This API includes a set of cryptographic functions that can be executed by the hardware coprocessors via requests from the OCSF cryptographic service provider to ICSF. OCSF also provides purely software cryptographic services.

- ▶ System SSL is the API used by a C/C++ client or server communicating over TCP/IP sockets and using the SSL protocol. System SSL uses ICSF services, if available. Otherwise, it uses its own software cryptographic services.

A fourth way of accessing the ICSF services is by using the BSAFE Toolkit 3.1 (or later). The BSAFE Toolkit is available from RSA Security Inc. on a license fee basis.

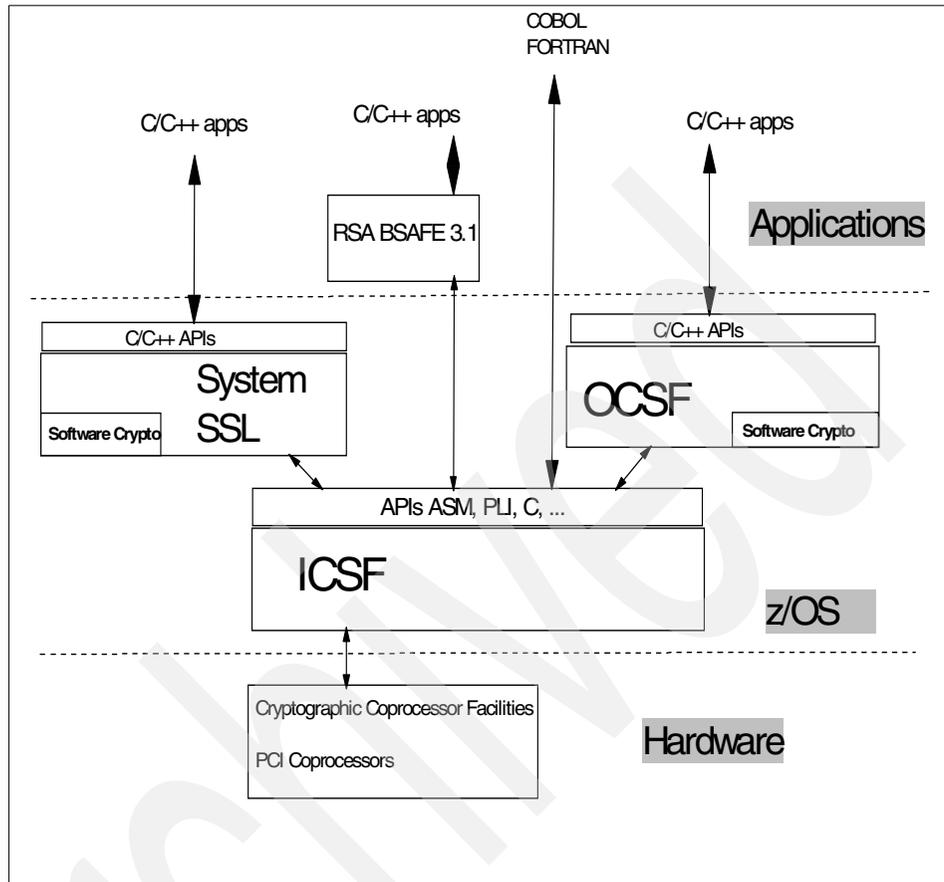


Figure C-1 Exploitation of the hardware coprocessors

## The IBM exploiters

In this section, we briefly describe the IBM software products that exploit zSeries hardware cryptography. In all cases, these exploiters use their own software cryptographic services if ICSF is not active on the system, or if they need cryptographic functions/algorithms that are not supported by ICSF (except for VTAM, which *does* require a cryptographic facility, either software or hardware, to be available to provide session-level encryption).

**Note:** The use of key length longer than 64 bits is still controlled to the extent that a specific FMID must be ordered for the z/OS components intended to use keys above this value.

The use of triple DES with hardware cryptographic assistance requires that both the correct component FMID and crypto enablement diskette be used during system setup.

## z/OS System SSL

Historically, the first SSL-enabled servers in OS/390 were developed before the availability of the OS/390 generic cryptographic services in System SSL (that is, before OS/390 V2R7), and had their own SSL code.

Since OS/390 V2R10, all SSL-enabled servers invoke the common System SSL DLLs provided in the base OS/390. Note that System SSL is usable both by SSL clients and servers running in OS/390.

Today, in z/OS V1R3, the SSL-enabled servers (which therefore use System SSL) are the following:

- ▶ IBM HTTP server
- ▶ CICS Transaction System
- ▶ TN3270 server
- ▶ FTP server and client
- ▶ LDAP server and client
- ▶ WebSphere Application Server

System SSL senses if CSF has been started in z/OS, and will use the hardware coprocessors for:

- ▶ Random number generation
- ▶ DES or Triple DES encryption and decryption, if DES or Triple DES are selected by the client and the server during the SSL handshake ciphersuites negotiation.

If RC2 or RC4 are selected, then System SSL performs encryption and decryption by software only.

- ▶ Encryption of the premaster secret at the SSL client, with the server's public key.
- ▶ Decryption of the premaster secret at the SSL server, with the server's private key.

System SSL comes with the GSKKMAN utility for managing keys and certificates in HFS key databases. GSKKMAN also calls for ICSF, if it has been started in the system.

The use of symmetric key lengths greater than 64 bits requires the installation of the System SSL Security Level 3 FMID.

**Note:** There is no way to explicitly enable or disable calls to ICSF by System SSL; System SSL will always use ICSF if it has been started in the system.

## z/OS Open Cryptographic Services Facility (OCSF)

OCSF in z/OS comes with three cryptographic service providers (CSPs). There are two software CSPs, which differ in the maximum key strength allowed for various symmetric and asymmetric encryption algorithms. And there is one hardware CSP: the IBM CCA Cryptographic Module.

An application that invokes the hardware CSP gets the following cryptographic services from ICSF:

- ▶ Digest MD5 and SHA-1 generation
- ▶ Random number generation
- ▶ DES or triple DES encryption and decryption
- ▶ RSA digital signature generate and verify
- ▶ RSA key pair generation (when a PCICC is available in the system)
- ▶ DES key exchange using RSA encryption
- ▶ Data encryption/decryption using RSA Optimal Asymmetric Encryption Padding (OAEP) algorithm (part of Secure Electronic Transaction (SET) protocol)

The use of key lengths greater than 64 bits requires the installation of the OCSF Security Level 3 FMID.

## IBM HTTP Server for z/OS

The IBM HTTP server for z/OS requires cryptographic services when requested to engage an SSL communication. It uses the System SSL DLLs, and therefore ICSF (if started in the system), beginning with OS/390 V2R10. Previous to OS/390 V2R10, IHS was delivered with its own SSL code.

If you intend to use symmetric keys longer than 64 bits, the IHS North American Secure FMID must be installed.

## **z/OS LDAP server and client**

The z/OS LDAP server or client requires cryptographic services when requested to engage an SSL communication. They use the System SSL DLLs, and therefore ICSF (if started in the system), beginning with OS/390 V2R7. Previous to OS/390 V2R7, the LDAP server and client were delivered with their own SSL code.

## **CICS Transaction Server and CICS Transaction Gateway**

The CICS transaction Server and Transaction Gateway require cryptographic services for their optional SSL communications.

- ▶ CICS Transaction Server is SSL-enabled at TS 1.3 and above. It uses the System SSL API.
- ▶ CICS Transaction Gateway calls the System SSL API, starting with version 3. Previous to version 3 it used its own, software-only, cryptographic services.

## **z/OS TN3270 Server**

The z/OS TN3270 server requires cryptographic services when requested to engage an SSL communication. It uses the System SSL DLLs, and therefore ICSF (if started in the system), beginning with OS/390 V2R10. Previous to OS/390 V2R10, the TN3270 server was delivered with its own SSL code.

## **z/os Firewall Technologies**

Three components of the z/OS Firewall Technologies use cryptographic services:

- ▶ IPSEC “tunnels”, which call ICSF (if active on the system) for DES or Triple-DES encryption and decryption. These calls are actually issued by the z/OS TCP/IP code. The Communications Server Security Level 3 FMID is required if triple-DES is to be used.
- ▶ The Firewall Technologies Configuration Server, which communicates with an optional GUI client workstation. The server and the client communicate using SSL and the server piece, on z/OS, uses System SSL.
- ▶ The Internet Security Association Key Management Protocol (ISAKMP) daemon, in OS/390 2.8 and above, is used to dynamically establish IPSEC tunnels secret keys, using the Internet Key Exchange (IKE) protocol. The two parties involved in an IKE negotiation can identify themselves, and initiate secure communications, using digital certificate and asymmetric and symmetric algorithms. The z/OS ISAKMP daemon calls OCSF for the related cryptographic services.

## z/OS DCE

z/OS DCE calls ICSF (if active on the system) for DES encryption or decryption, provided that the message length is greater than 60 bytes.

## z/OS Network Authentication Service (Kerberos)

z/OS NAS issues calls to ICSF for DES and triple-DES encryption/decryption beginning with z/OS V1R2. The use of triple-DES requires the installation of the Security Server Network Authentication Service Level 3 FMID.

## Payment Processing products

The IBM Payment Gateway™ for OS/390 and WebSphere Payment Manager support the use of either the SSL or SET protocol for secure transaction.

- ▶ When using SSL, the Payment Gateway calls System SSL for the required cryptographic services, whereas the Payment Manager uses its own (Java) SSL code.
- ▶ When using SET, the Payment Gateway detects the presence of an active ICSF instance that it will invoke for:
  - RSA digital signature generation and verification
  - Random number generation
  - DES encryption and decryption
  - OAEP block compose and decompose

The WebSphere Payment Manager or Payment Gateway utilization of hardware cryptography for the SET protocol is left to the user to decide, via a parameter in the SET configuration table (Payment Manager) or an environment variable (Payment Gateway).

Note that for the Payment Gateway, hardware cryptography support is required to get the SET gateway certification from the major card associations (Visa and MasterCard).

## VTAM Session Level Encryption

VTAM provides Session Level Encryption if a cryptographic product is available on the system; that is, VTAM does *not* provide its own cryptographic software. Historically, the Programmed Cryptographic Facility (PCF) software product provided the cryptographic services to VTAM before the availability of ICSF.

## RACF

Two components of RACF indirectly invoke external cryptographic services:

- ▶ The Open Cryptography Enhanced Plug-in (OCEP) is an OCSF Trust Policy and Data Library plug-in that allows you to work with the keys and certificates in the RACF Data Base. As such, it can request for OCSF cryptographic services.
- ▶ The RACDCERT RACF command permits you to generate RSA key pairs, using PCICC with z/OS V1R4, and to optionally store the private key into the PKDS, which is encrypted under the asymmetric Master Key. It also calls ICSF for certificate signature if the private key is stored in the PKDS.

## z/OS Public Key Infrastructure (PKI) Services

This z/OS Security Server component provides all the services expected from a PKI-compliant Certification Authority, beginning with z/OS V1R3. It uses ICSF through the RACF RACDCERT command.

## Crypto Based Transactions (CBT) banking solution

CBT is an IBM-specific solution to provide end-to-end security for transactions occurring between a client on the Internet and a z/OS transaction running as a batch or CICS program. It provides data confidentiality, integrity and non-repudiation. CBT on the z/OS end uses the Crypto Server Module (CSM) API, which relies on zSeries hardware cryptography.

## Java cryptography

The Java 2 platform-independent cryptography API is provided in the Java Cryptographic Extension (JCE) classes. As of IBM SDK 1.3.1, a new set of JCE classes known as IBMJCE4758 classes are an IBM implementation of JCE that invoke ICSF for the following cryptographic services:

- ▶ DES and triple-DES encryption and decryption
- ▶ MD2, MD5 and SHA1 hashing
- ▶ RSA signature generation and verification
- ▶ DSA signature generation and verification



## Crypto performance considerations

In this appendix, we discuss general considerations about the performance of cryptographic operations for zSeries cryptographic users. The intent is to introduce basic design and implementation concepts of cryptographic operations and the related performance implications.

Although there is some reference to the actual hardware implementation for zSeries cryptographic operations, specific implementations and actual performance data are beyond the scope of this redbook.

# General considerations for performance of cryptographic operations

All cryptographic operations used in digital cryptography can be performed in software, as the cryptographic algorithms are a sequence of logic operations on the data. The elementary operations used are, for example, permutations of the input data and XOR, with some other data as the cryptographic key.

A programmed sequence of the elementary operations of the cryptographic algorithm will yield the desired result. On the other hand, it is also conceivable that a programmed sequence of the cryptographic algorithm may be speeded up by special hardware that combines some of the elementary operations in fewer machine cycles, or that performs operations in parallel.

As cryptography in general has matured to the current level, various cryptographic algorithms have been developed to address various problems in cryptographic operations. This process is expected to continue into the future, as the protection of data by current algorithms is challenged by advances in processing speed of computer hardware, which may make “brute force attacks” tangible.

In considering the performance of cryptographic operations, we examine two major types of algorithms here in more detail: symmetric key algorithms and public key algorithms.

## ***Symmetric key algorithms***

In symmetric key algorithms, the same key is used to encrypt and decrypt the data to be secured. This type of encryption can be used to encrypt small or large quantities of data. Examples are Digital Encryption Standard (DES) or Advanced Encryption Standard (AES).

Key sizes are in the range of 8 to 32 bytes in length. Operation speed tends to be in the order of ten hardware cycles per unit of operation (typically 8 bytes or key length), if the special cryptographic operation can be executed in hardware. If the same algorithm is executed in software, the execution time tends to be five to ten times longer.

## ***Public key algorithms***

Public key algorithms are based on a pair of keys, one part being the private key (which has to be kept secure by the owner), and the other part being the public key (which is distributed in public). Examples are public key algorithms of Rivest Shamir Adleman (RSA) or Diffie Hellmann.

Key lengths are typically from 512 bits to 2048 bits (64 bytes to 256 bytes). The maximum data length that can be encrypted is limited to the length of the key.

The execution time (calibrated to the length of data ciphered) is in the order of ten times slower than symmetric key algorithms. Thus, public key algorithms are normally only used to securely transmit small amounts of data (for example, symmetric keys which are in turn used to encrypt the bulk data transfer).

Due to the speed difference between symmetric key and public key operations, and the different amount of data that can be handled, the zSeries cryptographic hardware implementation has chosen the following two different approaches:

- ▶ The hardware for *symmetric key operations* in most cases is operated synchronously to the CPU operations.

The CPU fetches and stores the data while, for many of the more commonly used algorithms, the cryptographic operations are executed in special hardware. The hardware has a fixed setup time per request and a fixed operation speed for the unit of operation. Thus, maximum throughput can be achieved for larger blocks of data, up to a hardware-defined limit.

- ▶ The hardware for *public key operations* works, in zSeries, mostly asynchronously to the CPU.

A CPU that needs to perform a public key operation uses a message queuing protocol to communicate with the public key hardware; after enqueueing a request to the hardware, the CPU will end this task and redispach another task. Thus, the public key cryptographic hardware will work in parallel to other tasks executed in the CPU. A special CPU task will poll for finished operations of the cryptographic hardware, dequeue them, and finally post the application waiting for the result of the cryptographic operation.

Several requests can be waiting in the queue to the hardware, either for execution or waiting for the result of the cryptographic operation to be dequeued by a CPU. Depending on the cryptographic hardware, one or several of the requests can be worked on in parallel.

As explained in more detail in Appendix C, “Exploitation of the cryptographic coprocessors” on page 317, all cryptographic hardware in zSeries systems can only be used through ICSF (which is a standard component of z/OS). ICSF is invoked by the application requesting a cryptographic service by an API. Thus, ICSF shields the complexity of the hardware communication from the user.

This ICSF operation service is an operating software path length which has to be added (from an application’s view) to the execution time of the cryptographic hardware. For symmetric key operations, it is advisable to have the application initiate a single API call for the *total* block of data, in order to limit the number of ICSF API calls.

Applications in the Internet environment will not use the cryptographic operations themselves; instead, they will follow protocol standards as, for example, Secure

Socket Layer (SSL). Executing the SSL protocol for a server (or client) on a zSeries system will result in a series of cryptographic operations which, under z/OS, invoke through ICSF either available cryptographic hardware or will be executed in software.

The SSL protocol will result in CPU path length (due to the protocol itself and due to ICSF), the symmetric key operations execution time (synchronous to the CPU), and the execution time of the public key operations (which may be parallel to the CPU operation).

It may be also be worth noting that supporting cryptographic operations for zSeries with hardware will not only improve performance, but also allows for better security than software encryption can offer. However, secure key cryptographic hardware operations may be slower than the equivalent clear key cryptographic hardware operations. Users will have to weigh security aspects versus performance in making a choice.

## RMF support for Crypto

Prior to z/OS V1R2, the performance impact of cryptographic processing was more or less transparent to capacity analysts. There was no information about the workloads using the cryptographic facilities, and to what extent they were used. The only indicator for cryptographic work was high utilization of the two processors extended with a crypto coprocessor.

This lack of information made it impossible to plan future capacity for additional cryptographic hardware. It was also not possible to solve performance problems immediately, since the causes of bottlenecks could not be identified easily.

Starting with z/OS V1R2, RMF adds more transparency in the area of cryptographic processing by reporting Crypto Using and Delay values in the Goals vs. Actuals section of the Postprocessor WLMGL report. Tasks that are found to be using or waiting for a CCF (synchronous or asynchronous) or a PCICC are reported on a service or report class period level.

In addition, RMF provides the new Postprocessor Crypto Hardware Activity report, which is based on new SMF records type 70 subtype 2. Besides detailed utilization information about each configured PCICC and PCICA card, the report provides performance measurements on selected ICSF activities on the CCF hardware.

While the data shown for PCICC and PCICA always reflects the total activity in a CPC, the data shown for CCF is for the partition. This functionality is available as SPE and needs to be installed as RMF APAR OW49808 (PTF UW99368) and

ICSF APAR OW51003 (PTF UW83664). Chapter 6, "Support functions" on page 241, describes the data reported in more detail.

When analyzing WLMGL report data and identifying certain work not achieving its goal (as indicated by a performance index greater than 1), high Crypto Using or Delay percentages are an indication of crypto hardware bottlenecks. At this point, detailed analysis of the Crypto Hardware Activity reports becomes advisable.

To determine which crypto hardware is assigned to a certain partition (MVS image), take a look at the hardware console. Each crypto hardware - CCF (at a maximum of 2), PCICC (at a maximum of 16) - has 16 queues and each of the queues can be assigned to a different partition. An individual queue number on every PCICC in the system relates to the same partition.

For PCICC, special attention should be given to RSA key-generation operations because these operations require a high amount of cryptographic processing capacity. Therefore, they are reported in addition to the total number of operations. The data for PCICA shows details for the two available algorithms, thus providing information about how the usage of these algorithms affects the utilization of the PCICA.





## TKE host TCP/IP server setup

This appendix describes the TCP/IP setup and customization required for proper TKE and TKE host communication.

### The main TCP/IP files to check and modify

At this point, we assume that your TCP/IP stack has already been installed and configured. This section discusses the parameters of interest for TKE communications.

**Note:** In the following examples, TCPIP has to be replaced by the high-level qualifier that you defined to your installation with the DATASETPREFIX statement in TCPI.DATA.

### TCPIP.HOSTS.LOCAL

This file contains TCP/IP hosts IP addresses and their corresponding domain names. This is the Site Table, which is intended to replace or to complement the services provided by the Domain Name Server from which you are requesting IP address resolution. Note that this information is exploited by the TKE TCP/IP

server for informational purposes only; messages in the server print file may be issued that mention the local host name and IP address.

As an example, following is the entry from our TCPIP.HOSTS.LOCAL file. Our TKE host is named MVN9 and has IP address 9.100.203.111.

```
HOST : 9.100.203.111 : MVN9 :::
```

## TCPIP.DATA

For standard servers and clients, the anchor configuration data set is the TCPIP.DATA data set. This is the main resolver configuration data set, with information on host name, domain origin, and so on.

In addition, it holds the TCPIPJOBNAME parameter (which identifies the TCP/IP stack to use) and the DATASETPREFIX parameter (which is used by the resolver code when allocating the other configuration data sets).

In our configuration, TCPIP.DATA contains these entries:

- ▶ HOSTNAME MVN9
- ▶ DATASETPREFIX TCPIP.OMVS
- ▶ TCPIPJOBNAME TCPIPOE

## TCPIP.PROFILE

The PORT statement in the TCPIP.PROFILE file is used to declare reserved ports, that is, ports that cannot be attributed as a result of a port request by any application. Only the application that runs with the jobname specifically indicated in the PORT statement can get access to the designated port.

In our configuration, we had the following entries in our TCPIP.PROFILE:

```
50003 TCP CSFTTCP ; TKE server
```

This entry indicates that port 50003 is reserved for the exclusive use of job CSFTTCP, which starts the TKE TCP/IP server in the TKE host system. It also indicates that port 50003 will be used for communications via the TCP transport layer.

The modifications made in the TCPIP.PROFILE are taken into account by stopping and restarting the related TCP/IP stack, or are made dynamically by using the VARY TCPIP,OBKEYFILE command. In order to bring a new PORT

dynamically, we entered the port statement to be dynamically executed in TCPIP.TCPPARMS(OBEYTK), as shown in Figure E-1.

```
BROWSE TCPIP.TCPPARMS(OBEYTK) - 01.02 Line 00000000 Col 001 080
***** Top of Data *****
;
-----
; Reserve ports for the following servers.
PORT
50003 TCP CSFTTCP ; crypto TKE server
***** Bottom of Data *****
```

Figure E-1 OBEYTK job

Then we issued the VARY TCPIP command followed by a D TCPIP,NETSTAT to check that the CSFTTCP port was actually dynamically reserved. (Since we had only one TCP/IP instance running in the system, we did not enter the TCP/IP task name in the commands; see Figure E-2.)

```
TCPIP, ,OBEYFILE,TCPIP.TCPPARMS(OBEYTK)
EZZ0060I PROCESSING COMMAND: VARY TCPIP, ,OBEYFILE,TCPIP.TCPPARMS(OBEYTK)
EZZ0300I OPENED OBEYFILE FILE 'TCPIP.TCPPARMS(OBEYTK) '
EZZ0309I PROFILE PROCESSING BEGINNING FOR 'TCPIP.TCPPARMS(OBEYTK) '
EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE 'TCPIP.TCPPARMS(OBEYTK) '
EZZ0053I COMMAND VARY OBEY COMPLETED SUCCESSFULLY
D TCPIP, ,N,PORTL
EZZ2500I NETSTAT CS V2R8 TCPIPOE 637
PORT# PROT USER FLAGS RANGE
00020 TCP OMVS D
00021 TCP OMVS DA
00023 TCP OMVS DA
00080 TCP OMVS DA
00111 TCP OMVS DA
50003 TCP CSFTTCP DA
```

Figure E-2 Vary TCP/IP command

**Note:** The change resulting from the OBEYFILE command is not carried across stopping and starting of the TCP/IP instance; therefore, you must edit the TCPIP.PROFILE in order to permanently install the change.

## TKE Host Transaction Program installation

The Host Transaction Program is the interface between the TKE Workstation and the crypto coprocessors. It includes the following software components:

- ▶ The CSFTTCP started procedure, which invokes the Terminal Monitor Program which, in turn, executes the CSFTHTP3 REXX exec
- ▶ The CSFTHTP3 REXX exec, which is a member of CSF.SCSFCLIO (where CSF can be another high-level qualifier specific to your installation).
- ▶ A parameter file CSFTPRM, used by the CSFTHTP3 exec
- ▶ The CSFTTKE module in the CSF.SCSFMOD0 library
- ▶ The so-called “Crypto Module (CM) Data Set”, which is used to maintain information about the crypto coprocessors in the configuration

## CSFTTCP started procedure installation

CSFTTCP can be copied from SYS1.SAMPLIB into the procedure library of the installation and customized according to your installation requirements. In our installation we decided to implement the parameter file CSFTPRM as a member of our SYS1.PARMLIB.MVN9 data set, as follows.

```
CSFTTCP PROC LEVEL=SYS1, MEMBER=CSFTHTP3,
//          CPARM=' '
//CLIST    EXEC PGM=IKJEFT01,
//          TIME=1440,
//          PARM='EX ' &LEVEL. .SCSFCLIO (&MEMBER) ' ' &CPARM' ' EXEC'
//SYSPRINT DD SYSOUT=*
//SYSEXEC DD DSN=&LEVEL. .SCSFCLIO, DISP=SHR
//SYSPROC DD DSN=&LEVEL. .SCSFCLIO, DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD DUMMY
//TKEPARMS DD DSN=&LEVEL. .PARMLIB. &SYSNAME. (CSFTPRM), DISP=SHR
//*
//* CUSTOMIZE THE DSN TO BE TCPIP DATASET ON YOUR SYSTEM
//SYSTCPD DD DISP=SHR, DSN=TCPIP. TCPPARMS (TCPDATOE)
```

Figure E-3 CSFTTCP procedure

Execution of the TKE Host Transaction Program must occur under External Security Manager control (in our case, RACF). A new facility profile, CSFTTKE,

must be defined, and the user ID given to the started task must be permitted to the profile.

We created a user ID TKEUSER with a TSO segment but, for security purposes, with the NOPASSWORD attribute; that is, it is a “protected” user. (Note that we do not show the creation of the user catalog or any other specific user-related facilities that may be required by your installation policy.)

In our case, we also had to permit TKEUSER to SYS1.PARMLIB to enable the task to read the CSFTPRM file; see Figure E-4.

```
TKEUSER NOPASSWORD TSO(ACCTNUM(AX0000) PROC(IKJSYS) UNIT(SYSDA))
ADDS 'TKEUSER.**' OWNER(SYS1) UACC(NONE)
PE 'SYS1.**' ACC(READ) ID(TKEUSER)
RDEFINE STARTED CSFTTCP.* STDATA(USER(TKEUSER))
SETROPTS RACLIST(STARTED) REFRESH
RDEFINE FACILITY CSFTTKE UACC(NONE) OWNER(SYS1)
PERMIT CSFTTKE CLASS(FACILITY) ID(TKEUSER)
SETROPTS CLASSACT(FACILITY)
SETROPTS RACLIST(FACILITY) REFRESH
```

*Figure E-4 Listing of TKEUSER*

An additional level of security can be implemented by defining the APPL class profile CSFTTKE in RACF, and by permitting to this profile the approved TKE users only (as individual RACF user IDs or as a RACF group).

```
SETROPTS CLASSACT(APPL)
SETROPTS RACLIST(APPL)
RDEFINE APPL CSFTTKE UACC(NONE)
PERMIT CSFTTKE CLASS(APPL) ID(userid or group) ACCESS(READ)
SETROPTS RACLIST(APPL) REFRESH
```

*Figure E-5 RACF APPL class list*

Note that if the RACF CSFSERV general resource class is active, the CSFTTCP user ID (TKEUSER, in our case) has to be permitted to CSFPCI and CSFPKCS.

```

IT CSFPKSC CLASS(CSFSERV) ACC(READ) ID(TKEUSER)
PERMIT CSFPCT CLASS(CSFSERV) ACC(READ) ID(TKEUSER)
SETROPTS RACLIST(CSFSERV) REFRESH

```

Figure E-6 CSFSERV Class active

Information on these profiles can be found in 6.1, “RACF access control to ICSF services” on page 242.

## The CSFTTKE module

The CSFTTKE command must be authorized in member IKJTSOxx of SYS1.PARMLIB (the CSFTTKE module resides in CSF.SCSFMODE0). The change to IKJTSOxx can be made dynamically by using the PARMLIB UPDATE(xx) command, where xx is the suffix of the modified IKJTSO member.

```

HCMD NAMES (          /***** AUTHORIZED COMMANDS *****/ +
CSFTTKE           /* AUTHORIZE TKE SERVER          */ +

```

Figure E-7 CSFTTKE command authorized

## The CSFTHTTP3 REXX exec

This exec resides in the CSF.SCSFCLI0 library. It is invoked by the CSFTTCP procedure and goes through an initialization phase before listening to the TKE request over the TCP/IP socket. During its initialization, CSFTHTTP3 reads the parameter file designated in the TKEPARMS DD card of CSFTTCP, that is, SYS1.PARMLIB(CSFTPRM).

CSFTPRM has to be edited to indicate what CM data set to use and which TCP/IP port to connect to. The default values are, respectively, &SYSNAME..TKECM and 50003. Our edited CSFTPRM is shown in Figure E-8 on page 337.

```

BROWSE      SYS1.PARMLIB.MVN9 (CSFTPRM) - 01.08          Line 00000000 Col 001 080
***** Top of Data *****
SET THE TKE DATA SETS; 'TKEUSER.TKECM'
PORT;50003
SET DISPLAY LEVEL;TRACE ALL
***** Bottom of Data *****

```

Figure E-8 Section of Port and CM dataset in CSFTPRM

#### Notes:

- ▶ We selected the TRACE ALL options. Other options are TRACE NON-ZERO, which traces non-zero return codes obtained during the transactions, and TRANSACTION TRACE, which traces transactions inputs and outputs. TRACE NON-ZERO is the default when no display level option is specified.
- ▶ We found that CSFTTCP fails if CSFTPRM contains sequence numbers in col 73 to 80. When editing CSFTPRM, ensure you have the sequence numbering on UNNUM.

## Starting the TKE Host Transaction Program

We recommend that you perform a preliminary check of TCPIP connectivity using a simple PING command from any workstation that can reach the TKE host TCP/IP stack.

When you issue the START CSFTTCP command, CSFTTHTTP3 is started and then listens for any incoming TCP/IP request.

Miscellaneous information, complemented with trace entries if tracing is active, is found in the SYSTSPRT DD data set, as shown in Figure E-9 on page 338.

```
THTP3 started at 19 Apr 2000, 09:50:54.
Generic user id.:      TKEUSER
CM dataset:           'TKEUSER.TKECM'
RACF environment:     "Cipher"
TCP/IP port:          50003
Display level         "Trace all"
=====
19 Apr 2000 09:50:55 SocketSetList: ReturnCode=0, SocketSetList: ''
Misc. host info:
  REXX/SOCKETS version = REXX/SOCKETS CS V2R6 APR 17,1998
  Domain name          =
  Host Id (ipaddress)  = 9.100.203.111
  Host Name            = MVN9
```

Figure E-9 CSFTTCP startup

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 340.

- ▶ *IBM eServer zSeries 900 Technical Guide*, SG24-5975

## Other resources

These publications are also relevant as further information sources:

- ▶ *TKE Workstation User's Guide 2000*, SA22-7524
- ▶ *zSeries Support Element Operations Guide*, GC38-0608
- ▶ *PR/SM Planning Guide*, GA22-7236
- ▶ *Security Server (RACF) Security Administrator's Guide*, SC28-1915
- ▶ *ICSF Administrator's Guide*, SA22-7521
- ▶ *ICSF System Programmer's Guide*, SA22-7520
- ▶ *ICSF Application Programmer's Guide*, SA22-7522
- ▶ *Maintenance Information for Desktop Consoles*, GC38-3115
- ▶ *OS/390 Cryptographic Services Secure Sockets Layer Programming Guide and Reference*, SC24-5877

## Referenced Web sites

These Web sites are also relevant as further information sources:

- ▶ All FIPS 140-1 products and their associated certification levels:  
<http://csrc.nssl.nist.gov/cryptval/140-1/1401val.htm>
- ▶ Detailed information on the SSL protocol Netscape:  
<http://home.netscape.com/security/>

- ▶ For detailed information on the 4758 Technology code segments concepts and mechanisms, check the IBM 4758 PCI Cryptographic Coprocessor site:  
<http://www.ibm.com/security/cryptocards>

## How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

## IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

# Index

## A

- access control points 311
- Adjunct Processor 22, 24
  - AP ID 22
  - management 37
- Advanced Facilities 97
- AES encryption 126
- AES encryption/decryption 126
- AES support 126
- AIX 2
- Apache HTTP server 277
- Apache Web server 278
- APAR OW46381 311
- asymmetric-keys 128
- ASYM-MK 128
- authority definition 97

## B

- Battery 34
- BSAFE 6–7
  - Toolkit 318

## C

- C/C++ client 318
- CCA 3, 41
  - Key Management Concepts Implementation 4
  - Key management functions 2
- CCF 16, 22, 46
  - Access Control page 220
  - Authorities page 216
  - Co-sign 238
  - Domains page 225
  - Error Handling for Cryptographic Coprocessor Feature (Subtype 4) 244
  - notebook 191
  - The ten CCF commands 222
- CDMF 127
- CEC 22
- CHPID 45
- CICS Transaction Server and CICS Transaction Gateway 322
- CKDS 8

- CKDS functions 126
- CKDS Refresh (Subtype 8) 245
- Dynamic CKDS Update 245
- CMOS 5
  - CMOS technology 5
  - cryptographic coprocessor 16
- Code-Signing Key 43
- Control domain index 120
- Control Vector 3
- CP/Q 41
- CPC 115, 123
  - crypto module 97
  - Crypto page 118
  - Cryptographic Key Data Set 8
  - Cryptography hardware implementation on z800 20
- CSFSERV 242
- CSFTHTP3 REXX exec 336
- CSFTPRM 334
- CSFTTCP 334
  - port 333
  - procedure 336
- CSFTTCP started procedure 334
- CSFTTKE 336
- CSFTTKE module 336
- CSM (Crypto Server Module) 324
- CV 3
- CW/CVC 6

## D

- Data Confidentiality 2
- Data integrity 2
- DCE 323
- Dedicated central processors 117
- DES 5, 8, 127
  - DES key 127
- Diffie-Hellman 8
- digital signature 6, 8
- Digital Signature Standard (DSS) 47
- diskette 44
- Domain 9, 97, 123
- Double Key MAC 6
- DSS 5

## E

- EBCDIC characters 142
- Enable cryptographic functions 120
- Enable Integrated Cryptographic Facility (ICRF) key entry 121
- Enable modify authority 121
- Enable public key algorithm (PKA) facility 119
- Enable Public Key Secure Cable (PKSC) and the integrated cryptographic service facility 121
- Enable query signature controls 121
- Enable query transport controls 121
- Enable special security mode 120
- ESCON 19

## F

- FCV 9, 40, 103
  - diskette 44
  - FCV enablement 44
  - import 104
  - overview 103
- Feature Code 55
  - feature code 0875 17
- FICON 19
- FIPS-4 33
- FRU 35
- Function Control Vector 55

## G

- GSKKYMAN 321

## H

- Hard Tamper 33
- hardware 55
- Hardware status display 130
- hashing algorithms 5
- HSA 37, 45

## I

- IBM 4758 card 16
- IBM 4758 CCA application 40
- IBM 4758-2 cryptographic coprocessor 23
- IBM HTTP Server for z/OS 321
- IBM Integrated Cryptographic Facility 2
- IBM Transaction Security System 2
- IBM Vital Product Data (VPD) database 50
- ICAZ90STATU 260
- ICRF 6

## ICSF 2, 8–9

- callable service and the service stub 306
- changes from previous release 125
- functions 6
- ICSF instance 10
- ICSF-managed VSAM data sets 8
- ICSF-owned data space 8
- ISPF panels 11
- overview 47
- PCICC Management panel 90
- RACF access control to ICSF services 242
- setup 124
- software 22
- TSO panels 126
- Utilities panel 136

## ICSF/MVS 2

- IFL (Integrated Facility for Linux) 24
- IKE (Internet Key Exchange) protocol 322
- Image Profile 117
- IML 39
- Import 45
- IMP-PKA keys in the workstation key storage 98
- Initial Master Key entry with the pass phrase initialization utility 128
- Integrated Cryptographic Services Facility/MVS 2
- Intrusion Latch 34
- IO Supervisor 46
- IOCDs 90
- IPSEC 322

## J

- Java Cryptographic Extension (JCE) classes 324
- Java Cryptography 324
- JCE classes 324

## K

- Key Management 2
- Key Separation 3

## L

- LDAP server 24
- libica API 263
- LIC 9
- Linux 255
  - I/O interface 256
- logical crypto coprocessor 117
- logical partition 118

LPAR 101  
  setup 112  
LPAR domains and TKE 9  
LPAR image 78

## M

Master Key 128  
Master Key Concept 3  
Master Keys 127  
  changing the PKA master Keys via ICSF 132  
  DES Master Key 127  
  PKA key Management Master key 127  
Master keys  
  PKA Signature Master Key 127  
MasterCard 47  
Memory Bus Adapter (MBA) chips 91  
MES 35, 102  
MES upgrade 96  
Miniboot 41  
MVN9 332

## N

Netscape 339  
NOPASSWORD 126

## O

OBEYTKE 333  
OCSF 322  
Online List 121  
Open Cryptography Enhanced Plug-in (OCEP) 324  
OpenSSL  
  Building OpenSSL with ibmca engine included  
  273  
  OpenSSL Functions 264  
  patches 272  
  Testing with the OpenSSL engine 276  
OPK 5  
Option Data Set 10  
OS/390 2  
OS/390 R9 103  
OS/390 V2R9 23  
OS/400 2  
OSA-Express 19

## P

PCI Crypto Accelerator card 30  
PCI Cryptographic Accelerator Feature 7

PCI Serial Number 80  
PCICA 7  
  data flow 30  
  features 6  
  PCICA cards 106  
  PCICA enablement 74  
PCICC 23, 101  
  ASYM-MK 125  
  card  
    physical security, handling, and shipping 32  
  code structure and UDX 296  
  Co-Sign page - 213  
  Create a PCICC new authority 205  
  Delete Authority 207  
  Domain Keys page 209  
  FCV diskette 256  
  Generate PCICC signature keys 203  
  modules notebook 191  
  notebook AUTHORITIES pages 203  
  PCICC and cryptographic domains 122  
  PCICC card 26  
  PCICC data flow 25  
  PCICC initialization 38  
  PCICC microcode load 40  
  PCICC microcode patches 43  
  PCICC removal 84  
  Roles 188  
  PCI-CC and PCI-CA viewing status 28  
  PCICC notebook DOMAINS page 36  
  Personal Authentication 2  
  PING 337  
  PKA 119, 130  
  PKA Key management 4  
  PKCS#11 API 263  
  PKDS 8, 125  
    Dynamic PKDS update 247  
    PKDSCACHE 125  
    Re-encipherment 125  
PORT 332  
Power On 39  
Power-on Reset (POR) 18, 39  
PPINIT 129  
PR/SM 10, 97, 114  
  PR/SM definitions 11  
  PR/SM environment 114  
  PR/SM microcode 117  
  PR/SM panels 125  
  setup 97  
Private Key Data Set 8

PSC 98  
Pseudo Random Number Generation 5  
Public Key Algorithm 119  
Public Key Algorithm (RSA) 6

## Q

Quiescing 262

## R

RACF 334  
    CSFKEYS class 242  
    CSFPKDR 242  
    CSFSERV 242  
    New profiles in the CSFSERV class 242  
Redbooks Web site 340  
    Contact us xiii  
RMF  
    reporting 251  
ROM 43  
RSA 5–7, 40  
    decryption 283  
    key token 280  
    operations 264  
    private key 261  
    RSA Optimal Asymmetric Encryption Padding (OAEP) 321  
    RSA Private Key Decryption 259  
    RSA Private Key Encryption 259  
    RSA Public Key Decryption 259  
    RSA Public Key Encryption 259  
    RSA signature 259

## S

Sales Ordering System 56  
SE 39  
SecureWay 5  
segment 43  
SET 6  
signature key 97  
Single-signature commands 195  
SMF 243  
    record type 70 249  
    record type 72 250  
    record type 82 243  
Soft Tamper 34  
Special Security Mode (SSM) 120  
SSL

    application example 263  
    SSL clients 320  
    SSL DLLs 320  
SSL (Secure Sockets Layer) protocol 16  
STI controller card 26  
Support Element 38, 46, 114  
SuSE SLES-7 (s390) - Kernel .4.7-SuSE-SMP 266  
symmetric-keys 127

## T

TCP/IP 47, 331, 337  
    socket 336  
    TCP/IP host for the TKE 120  
TCP/IP file  
    TCPIP.HOSTS.LOCAL 331  
TCP/IP files 331  
    TCPIP.DATA 332  
    TCPIP.PROFILE 332  
TDES 78  
Thermal container 37  
thermal gel 37  
TKE 11, 15, 101, 128  
    Authority signature key indicator 172  
    Customize TKE.INI parameters 168  
    definitions 154  
    Host definitions 97  
    Host system S/390 definition and logon from TKE 176  
    Host Transaction Program 334  
    Host Transaction Program installation 334  
    Introduction to the TKE V3.1 workstation 148  
    Loading the first part of the TKE Master Key 166  
    logon 173  
    Major changes 148  
    Master key 156  
    OS/2 desktop 156  
    Starting the Application 169  
    TKE V3.1 software 153  
    TKE Workstation access control 154  
    users 316  
    workstation installation 153  
    workstation TCP/IP setup 156  
TKE Workstation 8  
Toggle 104  
Triple DES 5  
TSO 47, 335

## U

UDX 41  
    building the UDX coprocessor executable 304  
    callable service 299  
    code 296  
    Development Process 302  
    file activation 90  
    image name 90  
    Invocation 298  
    management 101  
    related definitions in the OPTIONS Data Set  
    141  
    UDX Configuration panel 111  
    UDX enablement 105  
    UDX installation 84  
    UDX status 104  
Usage domain index 119  
User Defined Extensions 41  
User defined Function (UDF) 296  
User Developed Program (UDP) 296

    internal structure 91  
    z900 channel subsystem 91  
z90crypt  
    API 260  
    driver 257  
    z90crypt-2.4.7-s390-1.tar.gz 266  
zeroize 36

## V

VBAT 33  
Visa 47  
Visa/MasterCard CW/CVC card-verification 6  
VM 265  
VTAM 319

## W

WebSphere Payment Manager 323

## X

X-ray exposure 33

## Z

z/OS Firewall Technologies 322  
z/OS ISAKMP 322  
z/OS Network Authentication Service (Kerberos)  
323  
z/OS Open Cryptographic Services Facility 321  
z/OS System SSL 320  
z/OS TCP/IP Host Transaction Program 160  
z/OS TN3270 Server 322  
z/VM 12  
    Crypto Support 11  
z800 23  
z900 91





## zSeries Crypto Guide Update

(0.5" spine)  
0.475" <-> 0.875"  
250 <-> 459 pages







# zSeries Crypto Guide Update



## **PCICC and PCICA installation and customization**

## **Linux for zSeries Crypto support**

## **UDX implementation on zSeries**

This IBM Redbook is designed to help you understand and implement the z/OS Cryptographic PCICC and PCICA cards. Although this book focuses on the enablement of the z/OS PCICC and PCICA products, cryptography and the available services on z/OS are also discussed and explained, with special attention given to the new Trusted Key Entry (TKE 3.1) Workstation.

In addition, we look at how Linux for zSeries supports the exploitation of the PCICC and PCICA cryptographic coprocessors by using a generic device driver called z90crypt, which routes the cryptographic work to the PCICC or PCICA cards.

We also describe User Defined Extensions (UDX), which is the facility offered by the S/390 and zSeries PCICC, when running under control of ICSF, to enable users to design and implement their own cryptographic services to be executed in the PCICC itself. This provides maximum flexibility to the PCICC user, along with all the protection that the card can offer.

## **INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

## **BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)