MAINFRAME CRYPTO

Unscrambling the Complexity of Crypto!

Securing Your Crypto Infrastructure

Greg Boyd

(gregboyd@mainframecrypto.com)

THE Z EXCHANGE

June 2018

Copyrights and Trademarks

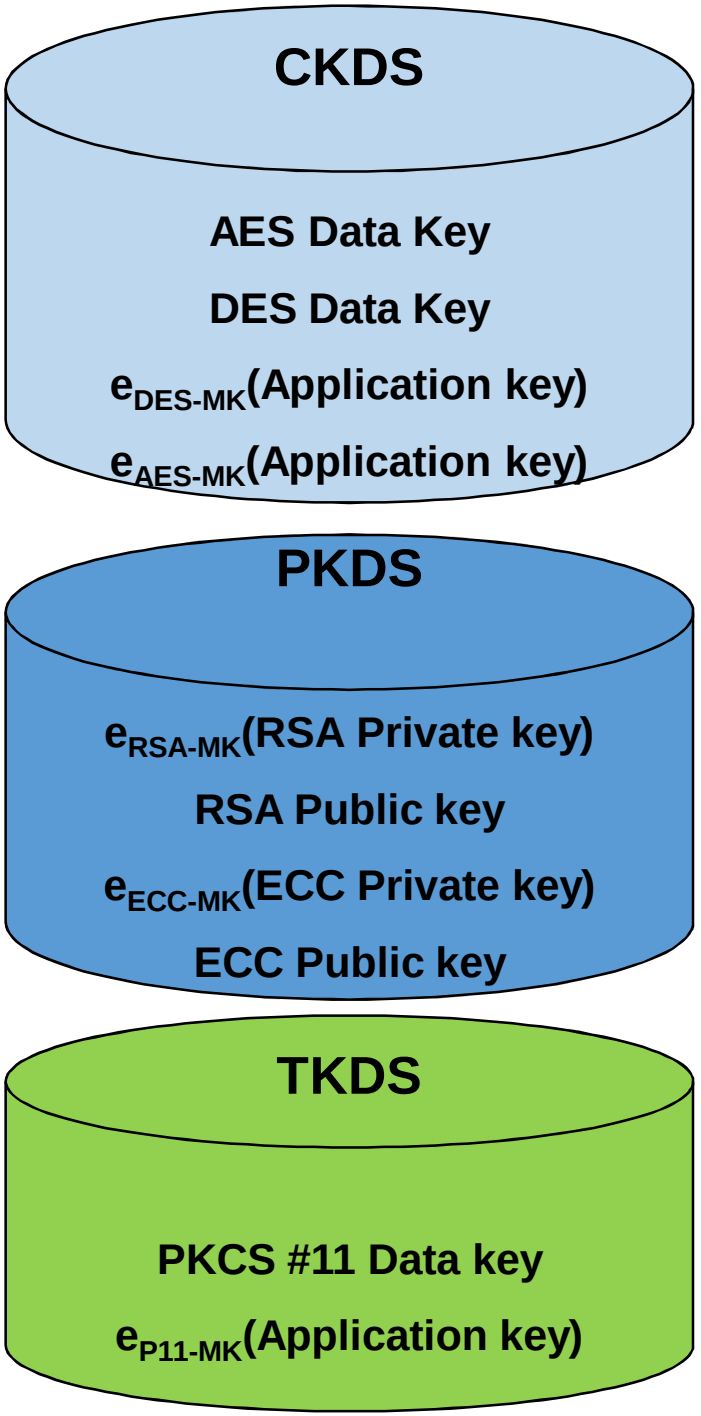
Agenda

- Securing crypto datasets (keystores, libraries)
- Securing ISPF Access
- ICSF Options (that impact security)
- Securing crypto resources
  - Keys
  - Services
    - ICSF Panels
    - APIs
    - Key Generation Utility Program
  - XCSFKEYS (secondary keys)
  - CRYPTOZ (PKCS #11 Tokens)
- Keystore policies (XFACILIT Class)
- A couple more XFACILIT Class profiles
- Operator commands
- Securing master key parts

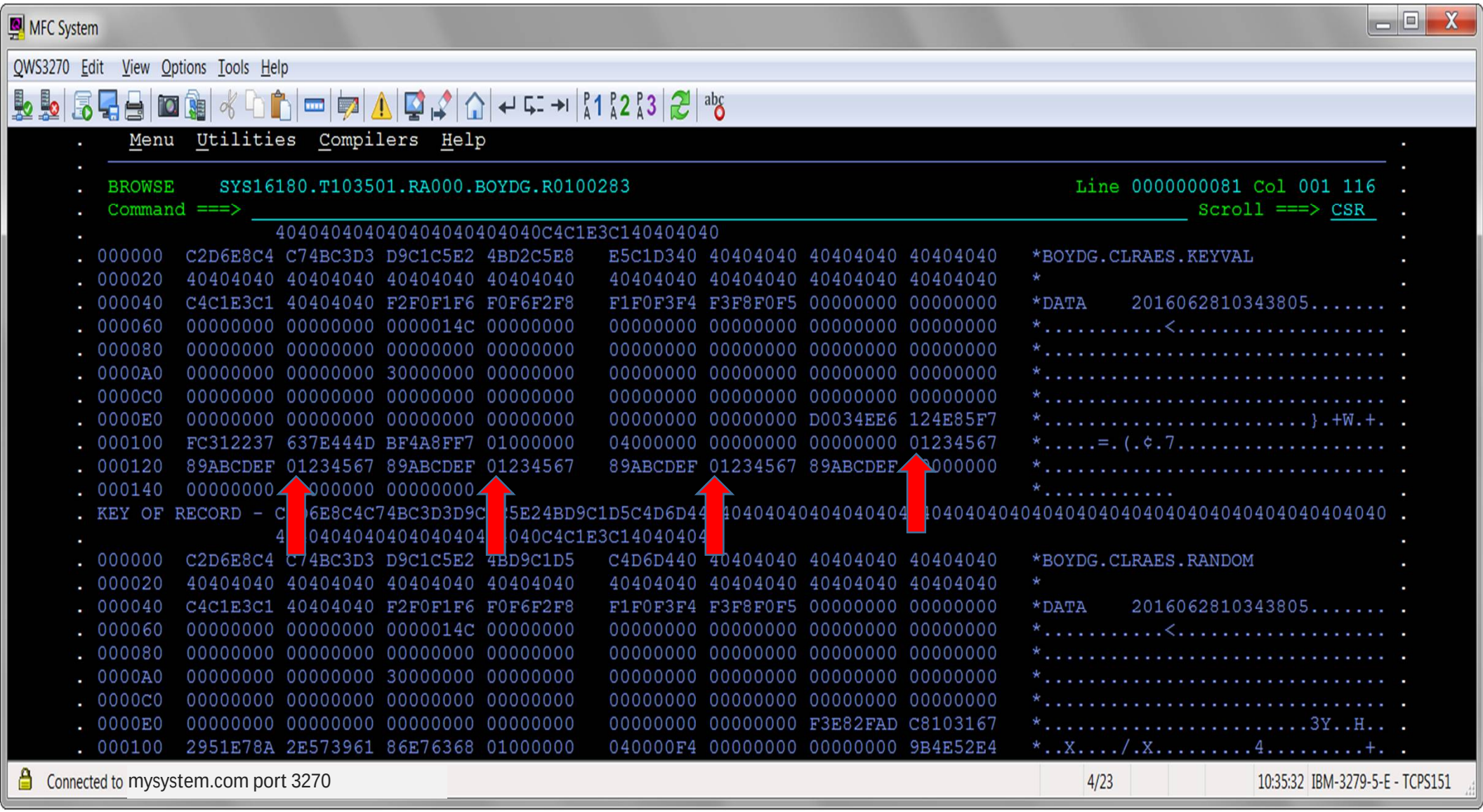
Key Repositories

- Cryptographic Key Data Set (CKDS)
  - Symmetric keys (AES, DES/TDES)
- PKA Key Data Set (PKDS)
  - PKA Keys (RSA, ECC)
  - Trusted PIN Blocks
- Token Key Data Set (TKDS)
  - Cryptographic Objects (AES, DES/TDES, ECC, RSA keys, all in a PKCS #11 format/architecture)

Secure the archives & backups too!



```
ADD TYPE(CLRAES),
        KEY(0123456789ABCDEF,0123456789ABCDEF,
                0123456789ABCDEF, 0123456789ABCDEF),
        LAB(BOYDG.CLRAES.KEYVAL)
```



Protect the ICSF Datasets

- CSFCLI0 – CLISTs
- CSFHDRS – Header File for C
- CSFMOD0 – Load Modules (APF Authorized)
- CSFMOD1 – Load Modules
- CSFMSG0 – ISPF Panel Messages
- CSFOBJ – Data (for compiling and link editing)
- CSFPNL0 – Panel Library
- CSFSKL0 – ISPF Skeletons
- CSFSTUB – Callable Services Stubs
- CSFTLIB - Tables

ISPF Access

Do you want everyone to be able to access the ICSF panels? Probably not …

- Allocate ICSF ISPF Libraries
  - Via TSO Logon PROC
  - Via REXX EXEC/CLIST and LIBDEF
- Invoke ICSF
  - Panel option on menu
  - Via REXX EXEC/CLIST

CSFSERV – Protect panels and utilities

Service	Description	Service	Description
CSFBRCK	CKDS Browser	CSFPMCI	Pass phrase master key/KDS initialization (TSO panel) utility
CSFCMK	Change master key (TSO panel) utility	CSFREFR	Refresh CKDS (TSO panel) utility
CSFCRC	Coordinated KDS Administration	CSFRENC	Reencipher CKDS (TSO panel) utility
CSFDKCS	Master key entry (TSO panel)	CSFRSWS	Administrative control functions (TSO panel) utility (ENABLE)
CSFGKF	Generate key fingerprint	CSFRWP	CKDS Conversion2 – rewrap option
CSFKGUP	Key Generation Utility Program	CSFSMK	Set master key utility
CSFOPKL	Operational key load	CSFSSWS	Administrative control functions (TSO panel) utility (DISABLE)
CSFPCAD	Cryptographic processors management (activate/deactivate)	CSFUDM	User Defined Extensions (UDX) management functions
CSFPKDR	PKDS reencipher and PKDS activate (TSO panel) utilities		

ICSF Options

- SSM(YES/**NO**) – Special Secure Mode
- CHECKAUTH(YES/**NO**) – Bypass SAF checks for supervisor state/system key callers
- FIPSMODE(YES/**NO**/COMPAT,FAIL(YES/NO))
  - YES - Enforces FIPS 140-2 modes for PKCS #11 APIs
  - COMPAT – Enforces FIPS 140-2 modes for PKCS #11 APIs based on FIPSEXEMPT.token_name
    - No profile or profile exists but caller has access NONE – FIPS algorithms and keysizes are enforced
    - Profile exists, caller has access READ (or higher) – FIPS algorithms and keysizes are not enforced
  - NO- FIPS 140-2 modes are not enforced
  - If running YES or COMPAT, then the FAIL option controls whether ICSF will stop if a FIPS environment can not be established
    - FAIL(YES) – Stop ICSF because it can't guarantee that a weak algorithm won't be used
    - FAIL(NO) – Allow ICSF to come up and do work, but some apps might fail if weak algorithms or keys are used, depending on the reason the FIPS environment could not be established

Protecting ICSF Resources

- CSFKEYS Class – controls access to CCA cryptographic keys
- CSFSERV Class – controls access to
  - CCA and PKCS #11 services
  - ICSF TSO panel utilities
  - KGUP
- XCSFKEY Class – controls authorization checks when symmetric key export services are used
- XFACILIT Class – to implement key store policies
- CRYPTOZ Class - controls access to and defines policy for crypto info within PKCS #11 tokens

CSFKEYS - Key Labels

- ICSF Key label
  - 64 byte character string, left justified, right padded with blanks
  - 1st character alphabetic or national (#, $, @)
  - The rest can be alphabetic, national or period (.)
  - All alphabetic are upper case
- Bad Examples
  - THISISAREALLYLONGKEYLABEL
  - MYKEY
  - X
- Good Examples
  - hlq.application.algorithm.date
    - PROD.APPX.AES56 .D180321
  - hlq.application.subsys.date.identifier.cycle
    - BOYDG. APPX.DB2.D180316. PAYROLL.V1

CSFKEYS Access

- RDEFINE CSFKEYS ** UACC(NONE)
- RDEFINE CSFKEYS PROD.* UACC(NONE)

- RDEFINE CSFKEYS PROD.APPX.DB2.*.PAYROLL.* UACC(NONE)
- PERMIT PROD.APPX.DB2.*.PAYROLL.* CLASS(CSFKEYS)
  ID(user) ACCESS(READ)

- RDEFINE CSFKEYS PROD.*.*.PE UACC(NONE)
- PERMIT PROD.*.*.PE CLASS(CSFKEYS) ID(groupid)
  ACCESS(READ)
  WHEN(CRITERIA(SMS(DSNENCRYPTION)))

Grouping Class

- GCSFKEYS – Resource Group Class for the CSFKEYS class
- GXCSFKEY – Resource Group Class for the XCSFKEY Class

Protected Key

- RALTER CSFKEYS PROD.APPX.DB2.*.PAYROLL.*
  ICSF(SYMCPACFWRAP(YES))

Protected Key (for Pervasive Encryption)

- RALTER CSFKEYS PROD.*.*.PE
  ICSF(SYMCPACFWRAP(YES) SYMCPACFRET(YES))

CSFSERV – Protecting ICSF APIs and functions

Resource Name	Callable Service Name	Callable Service Description
CSFENC	CSNBENC CSNEENC	Encipher
CSFDEC	CSNBDEC CSNEDEC	Decipher
CSFKGN	CSNBKGN CSNEKGN	Key Generate
CSFKGN2	CSNBKGN2 CSNEKGN2	Key Generate2
CSFPKRW	CSNDKRW CSNFKRW	PKDS Record Write
CSF1HMG	CSFPHMG CSFPHMG6	PKCS #11 Generate MAC
CSFIQF	CSFIQF CSFIQF6	ICSF Query Facility

See the ICSF Admin Guide, SC14-7506

CSFSERV – Protect panels and utilities

Service	Description	Service	Description
CSFBRCK	CKDS Browser	CSFPMCI	Pass phrase master key/KDS initialization (TSO panel) utility
CSFCMK	Change master key (TSO panel) utility	CSFREFR	Refresh CKDS (TSO panel) utility
CSFCRC	Coordinated KDS Administration	CSFRENC	Reencipher CKDS (TSO panel) utility
CSFDKCS	Master key entry (TSO panel)	CSFRSWS	Administrative control functions (TSO panel) utility (ENABLE)
CSFGKF	Generate key fingerprint	CSFRWP	CKDS Conversion2 – rewrap option
CSFKGUP	Key Generation Utility Program	CSFSMK	Set master key utility
CSFOPKL	Operational key load	CSFSSWS	Administrative control functions (TSO panel) utility (DISABLE)
CSFPCAD	Cryptographic processors management (activate/deactivate)	CSFUDM	User Defined Extensions (UDX) management functions
CSFPKDR	PKDS reencipher and PKDS activate (TSO panel) utilities		

RACF CSFSERV Profiles

- RDEFINE CSFSERV ** UACC(NONE) or
  RDEFINE CSFSERV CSF* UACC(NONE)

NOTE: New to ICSF do this now! If ICSF is already active, be careful!

- RDEFINE CSFSERV CSFCMK UACC(NONE)
- PERMIT CSFSERV CSFCMK
  ID(icsf admin userid or groupid) ACCESS(READ)

RACF CSFSERV Profiles (cont.)

- RDEFINE CSFSERV CSFKGN* UACC(NONE)
- PERMIT CSFSERV CSFKGN* ID(operational key officers group) ACCESS(READ)
  - Protects CSNBKGN and CSNBKGN2

- RDEFINE CSFSERV CSFSA* UACC(NONE)
- PERMIT CSFSERV CSFSA* ID(prod id) ACCESS(READ)
  - Protects CSNBSAD, CSNBSAD1, CSNBSAE and CSNBSAE1,

CSFSERV Access

- RACF
  - DEFINE CSFSERV * UACC(NONE)
  - RDEFINE CSFSERV service UACC(NONE)
  - PERMIT service CLASS(CSFSERV) ID(userid) ACCESS(READ)
- CA Top Secret
  - TSS ADD(owner) CSFSERV(service)
  - TSS PERMIT(userid/profile) CSFSERV(service) ACCESS(READ)
- CA ACF2
  - SET RESOURCE(SAF)
  - COMPILE
  - $KEY(service) TYPE(SAF) UID(userid) SERVICE(READ) ALLOW
  - STORE

CRYPTOZ Class – Users and Profiles

- PKCS #11 standard for systems that use PINs to grant access
- Two types of users
  - Standard User (user)
    - Access to private objects
    - Ability to change their own PIN
  - Security Officer (SO)
    - Can initialize a token
    - Set the User's PIN
    - Access public objects, but not private objects
- CRYPTOZ profile
  - USER.token-name – controls access of the user to the token
  - SO.token-name – controls access of the SO to the token

CRYPTOZ Class – Access Levels

- Three token access levels, defined by PKCS #11
  - User R/O – allows user to read token, including private objects, but cannot create new token or session objects or alter existing
  - User R/W – allows user read/write access to token objects, including private objects
  - SO R/W – allows user to act as security officer for the token and to read, create and later public objects
- Three token access levels, defined by z/OS
  - Weak SO – security officer can modify CA certificates contained in, but not initialize the token
  - Strong SO - security officer can add, generate or remove private objects in a token
  - Weak User – user that cannot change the trusted CAs in a token

CRYPTOZ Class – Token Access Levels

- Levels of Access
  - Read
  - Update
  - Control

CRYPTOZ resource	Read Access	Update Access	Control Access
SO.token-label	Weak SO (Can read, create, delete, modify and use public objects	SO R/W (Weak SO + create and delete tokens	Strong SO (SO R/W + read, but not use private objects; create, delete and modify private objects
USER.token-label	User R/O (Can read public and private objects	Weak User (User R/O + create, delete and modify private and public objects. Cannot add, delete or modify CA objects	User R/W (Weak User + add, delete and modify CA objects

Use means: Performing any crypto operating involving the key object; searching for key objects; retrieving sensitive key object attributes

Key Store Policies – XFACILIT Class

By defining a policy, a security administrator can make symmetric and asymmetric keys more secure, without requiring application changes

- Key Token Authorization Checking
- Default Key Label Checking
- Duplicate Key Token Checking
- Granular Key Label Access Control
- Symmetric Key Label Export Control (aka CSNDSYX Export Control)
- PKA Key Management Extension Control
- Key Archive Use Control

ICSF Application Key Usage

- Some APIs can use
  - Key ID – label of a key
    - 64-byte (padded to right) label
    - Must exist in the keystore
    - Key record contains the key token
  - Key token – CCA Key token
    - 64-byte structure as defined by CCA
  - Key material – actual key material
    - Appropriate length

Key Token Authorization Checking

Q: When an API is passed a key token, instead of a key label, how does ICSF validate whether the user is permitted to that key? There is no key label to use to find the profile!

A: With this check enabled, ICSF will search for that key material in the keystore and use the label from that record to search for a RACF profile.

- CSF.CKDS.TOKEN.CHECK.LABEL.action
- CSF.PKDS.TOKEN.CHECK.LABEL.action
  - Action
    - Fail – the use of the key token will fail
    - Warn – the use of the key token is allowed and a warning message is generated

RDEFINE XFACILIT CSF.CKDS.TOKEN.CHECK.LABEL.WARN

RDEFINE XFACILIT CSF.PKDS.TOKEN.CHECK.LABEL.WARN

Default Key Label Checking

Q: With the previous check enabled, what happens if there is no key record in the keystore that matches the token?

A: ICSF will permit the use of that key. However, you can override, by enabling Default Key Label Checking.

Provides a default key label and thus a default key label profile for SAF checks

- CSF.CKDS.TOKEN.CHECK.DEFAULT.LABEL
  - CSF-CKDS-DEFAULT
- CSF.PKDS.TOKEN.CHECK.DEFAULT.LABEL
  - CSF-PKDS-DEFAULT

```
RDEFINE CSFKEYS CSF-CKDS-DEFAULT UACC(NONE)
RDEFINE XFACILIT CSF.CKDS.TOKEN.CHECK.DEFAULT.LABEL
RDEFINE CSFKEYS CSF-PKDS-DEFAULT UACC(NONE)
RDEFINE XFACILIT CSF.PKDS.TOKEN.CHECK.DEFAULT.LABEL
```

Enforcing Keystore use

- ICSF does not require that key material be stored in the CKDS or PKDS.

- Enabling Key Token Authorization Checking plus Default Key Label Checking changes that behavior
  - Key Token Authorization Checking enforces that a copy of the key must be in the keystore
  - And Default Key Label Checking says there must be a label in the keystore that grants access

Duplicate Key Token Checking

Prohibit the defining of multiple copies of a key in the keystore

- CSF.CKDS.TOKEN.NODUPLICATES

  RDEFINE XFACILIT CSF.CKDS.TOKEN.NODUPLICATES

- CSF.PKDS.TOKEN.NODUPLICATES

  RDEFINE XFACILIT CSF.PKDS.TOKEN.NODUPLICATES

Enabling this security policy will not detect existing duplicate keys in the keystore

– use the CSFDUTIL utility to find dups

Granular Key Label Access Control

- By default, only READ authority to a key is needed to read from, create, write to, and delete a key (using the label)
- Granular Key Label Access Control increases the required authority
  - UPDATE authority required to create a key label
  - CONTROL authority required to update or delete a key

- CSF.CSFKEYS.AUTHORITY.LEVELS.WARN
  - A warning is issued as long as the user has read authority to the key label

- CSF.CSFKEYS.AUTHORITY.LEVELS.FAIL
  - The operation will fail unless the user has the appropriate authority above

```
RDEFINE XFACILIT CSF.CSFKEYS.AUTHORITY.LEVELS.WARN

PERMIT PE.AES256.KEY1 CLASS(CSFKEYS) ID(user) ACCESS(UPDATE)
PERMIT PE.AES256.KEY1 CLASS(CSFKEYS) ID(user) ACCESS(CONTROL)
```

Using a key vs Exporting a key

- Consider a key
  - Used to encrypt data
  - But also needs to be exported
  - Some users should have authority to one operation, but not the other
- CSFKEYS Class grants authority to use the key
- XCSFKEY Class grants authority for other operations

SETROPTS CLASSACT(XCSFKEY)

SETROPTS RACLIST(XCSFKEY)

CSNDSYX – export a key, under a transport key

- AES.PARTNER1.KEY1 – DATA key used to encrypt data that will be shared with Partner1, and this key must be securely transported to Partner1
- RSA.PARTNER1.TRANSPA – RSA key used to transport operational keys to Partner 1

```
CALL CSNDSYX( ....
        transporter_key_identifier=RSA.PARTNER1.TRANSPA
        source_key_identifier=AES.PARTNER1.KEY1,        ....)
```

- CSFKEYS RSA.PARTNER1.TRANSPA ACCESS(READ)
- CSFKEYS AES.PARTNER1.KEY1 ACCESS(READ)

Symmetric Key Label Export Control (aka CSNDSYX Export Control)

- CSF.XCSFKEY.ENABLE.AES

  RDEFINE XFACILIT CSF.XCSFKEY.ENABLE.AES

- CSF.XCSFKEY.ENABLE.DES

  RDEFINE XFACILIT CSF.XCSFKEY.ENABLE.DES

CALL CSNDSYX(….

transporter_key_identifier=RSA.PARTNER1.TRANSPA

source_key_identifier=AES.PARTNER1.KEY1, ….)

- CSFKEYS RSA.PARTNER1.TRANSPA ACCESS(READ)
- XCSFKEYS AES.PARTNER1.KEY1 ACCESS(UPDATE)

PKA Key Management Extension Control (Part 1)

- CSF.PKAEXTNS.ENABLE.WARNONLY
- CSF.PKAEXTNS.ENABLE

- Determines if a symmetric key can be exported, and which asymmetric key can be used to wrap it
  - ICSF(SYMEXPORTABLE(BYNONE))
  - ICSF(SYMEXPORTABLE(BYLIST) SYMEXPORTKEYS(...))
    - ADDSYMEXPORTKEYS(...)
    - DELSYMEXPORTKEYS(...)
    - SYMEXPORTCERTS(...)
  - ICSF(SYMEXPORTABLE(BYANY)

  CSFKEYS or XCSFKEYS!

Note: Requires an active key store policy

CSF.xKDS.TOKEN.CHECK.LABEL.action

CSF.xKDS.TOKEN.NODUPLICATES

PKA Key Management Extension Control (Part 2)

- CSF.PKAEXTNS.ENABLE.WARNONLY
- CSF.PKAEXTNS.ENABLE

- Determines how an asymmetric key can be used
  - ICSF(ASYMUSAGE(HANDSHAKE/NOHANDSHAKE))
  - ICSF(ASYMUSAGE(SECUREEXPORT/NOSECUREXPORT))

- By default ASYMUSAGE(HANDASHAKE SECUREEXPORT)

Key Archive Use Control

Security policy will allow an archived key to be used

- CSF.KDS.KEY.ARCHIVE.USE

```
RDEFINE XFACILIT CSF.KDS.KEY.ARCHIVE.USE
```

Key Store Policies – XFACILIT Class

- Key Token Authorization Checking[1]
- Default Key Label Checking[2]
- Duplicate Key Token Checking[1]
- Granular Key Label Access Control[3]
- Symmetric Key Label Export Control (aka CSNDSYX Export Control) [3]
- PKA Key Management Extension Control[2]
- Key Archive Use Control

- [1]Activates Key Store Policy
- [2]Requires an active key store policy
- [3]Does not require an active key store policy, unless you want to associate a key token with a label

A couple more in the XFACILIT class

- Dynamic SSM

  RDEFINE XFACILIT CSF.SSM.ENABLE

- Disable SAF checks for the hashing APIs

  DEFINE XFACILIT CSF.CSFSERV.AUTH.CSFOWH.DISABLE

- Disable SAF checks for the RNG APIs

  RDEFINE XFACILIT CSF.CSFSERV.AUTH.CSFRNG.DISABLE

Operator Commands

- SETICSF - MVS.SETICSF profile
  - Activate/Deactivate/Restart or Check/Delete devices
  - Enable/Disable keystores
  - Change/Refresh some ICSF Options
- Display ICSF - MVS.DISPLAY.ICSF profile
  - Information about available devices
  - Information about active keystores
  - Information about master key status
  - Information about members of the KDS Plex
  - Information about current options

RDEFINE OPERCMDS MVS.SETICSF UACC(NONE)

PERMIT MVS.SETICSF CLASS(OPERCMDS) ID(lead oper) ACCESS(UPDATE)

RDEFINE OPERCMDS MVS.DISPLAY.ICSF UACC(NONE)

PERMIT MVS.DISPLAYICSF CLASS(OPERCMDS) ID(operator group) ACCESS(UPDATE)

Securing Master Key Parts

```
CSFMKV00  ---------- ICSF –  Checksum and Verification and Hash Pattern ----------
COMMAND ===>

Enter data below:
 Key Type  ===>                                (Selection panel displayed if blank)
 Key Value ===>  02AC7633C1951F0A             Input key value 1
           ===>  5916A7A3DF8718DB             Input key value 2
           ===>  0000000000000000             Input key value 3 (AES, ECC & RSA Keys)
           ===>  0000000000000000             Input key value 4 (AES, ECC Keys only)

 Checksum        : DD                        Check digit for key value
 Key Part VP     : A7518C6F9C65FB02          Verification Pattern
 Key Part HP     : D8C18ADC8F01E6D9          Hash pattern
                 : 307C31F4CC1CB2F2

Press ENTER to process.
Press END     to exit to the previous menu.
```

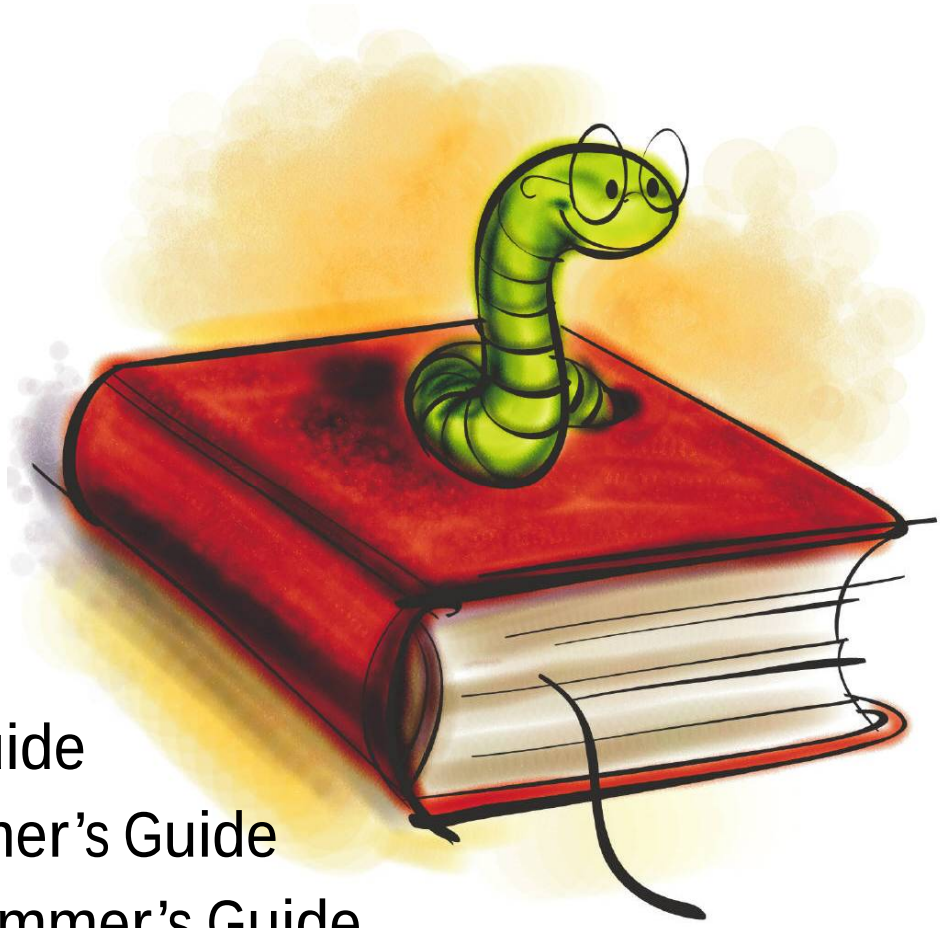
Trusted Key Entry Workstation

TCP/IP

Host w/ Crypto Express Coprocessor

Trusted Key Entry Workstation

ICSF Publications



- SC14-7505 ICSF Overview
- SC14-7506 ICSF Administrator's Guide
- SC14-7507 ICSF Systems Programmer's Guide
- SC14-7508 ICSF Application Programmer's Guide

Websites

- TechDocs (search on crypto)

http:/www.ibm.com/support/techdocs TechDoc TD105748

'CSFSERV Class RACF Profiles for ICSF Panels'

Questions …