MAINFRAME CRYPTO

Crypto Application Coding

Greg Boyd

gregboyd@mainframecrypto.com

March 2016

Copyrights . . .

- Presentation based on material copyrighted by IBM, and developed by myself, as well as many others that I worked with over the past 10 years

. . . And Trademarks

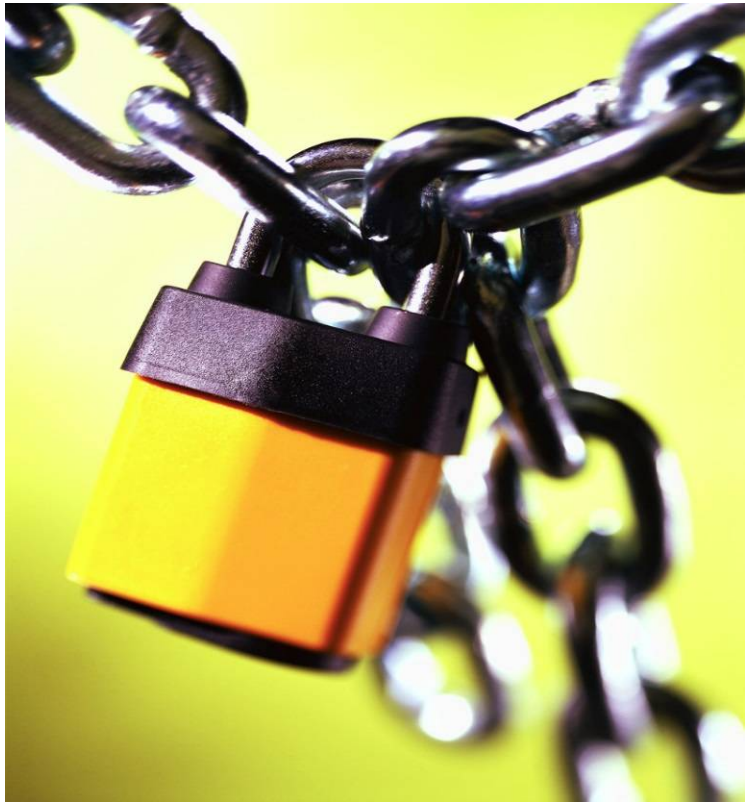
-
- All trademarks, trade names, service marks and logos referenced herein belong to their respective companies. IBM, System z, zEnterprise and z/OS are trademarks of International Business Machines Corporation in the United States, other countries, or both. All trademarks, trade names, service marks and logos referenced herein belong to their respective companies.
- **THIS PRESENTATION IS FOR YOUR INFORMATIONAL PURPOSES ONLY**. Greg Boyd and Mainframe Crypto, LLC assumes no responsibility for the accuracy or completeness of the information. TO THE EXTENT PERMITTED BY APPLICABLE LAW, THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. In no event will Greg Boyd or Mainframe Crypto, LLC be liable for any loss or damage, direct or indirect, in connection with this presentation, including, without limitation, lost profits, lost investment, business interruption, goodwill, or lost data, even if expressly advised in advance of the possibility of such damages.

Agenda – Application Coding

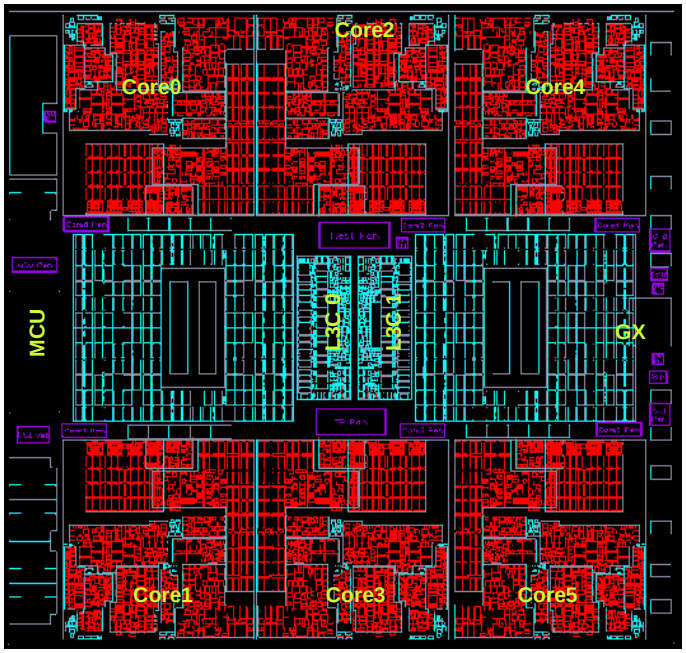
- Refresher
  - Crypto Functions
  - CPACF
  - Crypto Express cards
- Coding Native crypto instructions
- Coding APIs
  - API Naming Conventions
  - API Statistics
  - Clear key API
  - Secure key API

Crypto Functions

- Data Confidentiality
  - Symmetric – DES/TDES, AES
  - Asymmetric – RSA,Diffie-Hellman, ECC
- Data Integrity
  - Modification Detection
  - Message Authentication
  - Non-repudiation
- Financial Functions
- Key Security & Integrity



System z Crypto Hardware



CPACF

Crypto Express5S

Crypto Express4S

MSA – Message Security Assist

- MSA (z890/z990)
  - Cipher Message instruction (KM)
  - Cipher Message with Chaining instruction (KMC)
  - Compute Intermediate Message Digest instruction (KIMD)
  - Compute Last Message Digest instruction (KLMD)
  - Compute Message Authentication Code instruction (KMAC)
- MSA Extension 1 (z9) – added SHA-256, AES-128 and PRNG
- MSA Extension 2 (z10) – added AES-192, AES-256, SHA-512
- MSA Extension 3 (CEX3) – added Protected Key support
- MSA Extension 4 (z196)
  - Cipher Message With Cipher Feedback (CFB) instruction (KMF)
  - Cipher Message With Counter instruction (KMCTR)
  - Cipher Message With Output Feedback (OFB) instruction (KMO)
  - Perform Cryptographic Computation instruction (PCC)
  - Add AES Ciphertext stealing, GHASH; Enhance CMAC mode
- MSA Extension 5 (z13)
  - Perform Pseudo Random Number instruction (PPNO)

Assembler Instructions

- Cipher Message instructions
  - KM R1,R2
  - KMC R1,R2
  - KMF R1,R2
  - KMO R1,R2
  - KMCTR R1,R3,R2
- R1 pointer to output location
- R2 pointer to source location
- R2+1 length of source
- R3 pointer to a counter field
- Reg 0 contains Function Code
- Reg 1 pointer to parameter block

Function Codes

Bit 56 – Encrypt/Decrypt flag

Code	Function	Parm Block Size (Bytes)	Data Block Size (Bytes)
0	KM(C)-Query	16	-
1	KM(C)-DEA	8 (16)	8
2	KM(C)-TDEA-128	16 (24)	8
3	KM(C)-TDEA-192	24 (32)	8
9	KM(C)-Encrypted-DEA	32 (40)	8
10	KM(C)-Encrypted-TDEA-128	40 (48)	8
11	KM(C)-Encrypted-TDEA-192	48 (56)	8
18	KM(C)-AES-128	16 (32)	16
19	KM(C)-AES-192	24 (40)	16
20	KM(C)-AES-256	32 (48)	16
26	KM(C)-Encrypted-AES-128	48 (64)	16
27	KM(C)-Encrypted-AES-192	56 (72)	16
28	KM(C)-Encrypted-AES-256	64 (80)	16
50	KM-XTS-AES-128	32	16
52	KM-XTS-AES-256	48	16
58	KM-XTS-Encrypted-AES-128	64	16
60	KM-XTS-Encrypted-AES-256	80	16
67	KMC-PRNG	32	8

Parameter Block

KM/KMC/KMF/KMCTR/
KMO Query

Status Word 16-byte

KMC PRNG

Chaining Value 8-byte
Key 1 8-byte
Key 2 8-byte
Key 3 8-byte

KM

Cryptographic Key 8-, 16- or 24-bytes for DES/TDES 128-, 192- or 256-bits for AES
Wrapping Key VP 24-bytes for DES/TDES 256-bits for AES
XTS Parameter 128-bits

KMx Chaining

Chaining Value 8-, or 16-bytes for DES/TDES 128-, 192- or 256-bits for AES
Cryptographic Key 8-, 16- or 24-bytes for DES/TDES 128-, 192- or 256-bits for AES
Wrapping Key VP 24-bytes for DES/TDES 256-bits for AES

Some code

```
                L       R0,FUNCCODE
                LA      R1,CPARMBLK     R1 points to cipherparm block
                SR      R7,R7           zero Reg 7 for the length
                L       R7,INPUTLEN     R7 length of data
                LA      R6,INPUTDAT     R6 points @ data
                LA      R8,OUTPUTDA     R8 points to output field
DOKMC           DS      0H
                KMC     R8,R6           Do the encrypt
                BRC     1,DOKMC

FUNCCODE DC     XL4'00000014'   D'20' for Encrypt AES-256 bit
CPARMBLK  DS    0H
ICVVALUE  DC     XL8'B9D628DECEBA6B05'
          DC     XL8'C37892BD6134B48E'
KEYVALUE  DC     XL8'0B54388EDE6466BF'
          DC     XL8'9AF620DF135D47EC'
          DC     XL8'503406FFB8B1A9F6'
          DC     XL8'24F1A3BE76B26D16'
INPUTDAT  DC     XL24'My cleartext input msg '
INPUTLEN  DC     XL4'00000018'
OUTPUTDA DC      XL24' '
```

API Naming Conventions

This callable service prefix:	Identifies	
CSNBzzz / CSFzzz	31-bit	Symmetric Key Services and Hashing Services
CSNBzzz1 / CSFzzz1	31-bit ALET-qualified	
CSNEzzz / CSFzzz6	64-bit	
CSNEzzz1 / CSFzzz16	64-bit ALET-qualified	
CSNDzzz / CSFzzz	31-bit	Asymmetric Key Services
CSNFzzz / CSFzzz6	64-bit	
CSFPzzz	31-bit	PKCS #11 Services
CSFPzzz6	64-bit	
CSFzzz	31-bit	Utility Services and TKE Workstation Interfaces
CSFzzz6	64-bit	

From the ICSF Application Programmer's Guide

APIs

- Callable Services
  - Dependency on server type
  - Dependency on Crypto card
  - Languages: C, COBOL, PL/I, Assembler, Fortran (and REXX)
  - Problem vs Supervisor State
  - Task or SRB Mode
  - 64-Bit Access (as of HCR77A0)
  - Alternate entry points (ALET-qualified)

APIs

- Symmetric Key Management (48)
  - CSNBCKI – Clear Key Import
  - CSNBCVG – Control Vector Generate
  - CSNBCVT – Control Vector Translate
  - CSNBCVE – Cryptographic Variable Encipher
  - CSNBDKX – Data Key Export
  - CSNBDKM - Data Key Import
  - CSNBDCM – Derive ICC MK*
  - CSNBDSK – Derive Session Key*
  - CSNBDKG – Diversified Key Generate
  - CSNBDKG2 - Diversified Key Generate2
  - CSNBEDH – ECC Diffie-Hellman
  - CSNBGIM – Generate Issuer MK*
  - CSNBKEX – Key Export
  - CSNBKGN – Key Generate
  - CSNBKGN2 – Key Generate2
  - CSNBKIM – Key Import
  - CSNBKPI – Key Part Import
  - CSNBKPI2 – Key Part Import2
  - CSNBKYT – Key Test
  - CSBNKYT2 – Key Test2
  - CSBNKYTX – Key Test Extended
  - CSNBKTB – Key Token Build
  - CSNBKTB2 – Key Token Build2
  - CSNBKTR – Key Translate
  - CSNBKTR2 – Key Translate2
  - CSNBCKM – Multiple Clear Key Import
  - CSNBSKM – Multiple Secure Key Import
  - CSNDPKD – PKA Decrypt
  - CSNDPKE – PKA Encrypt
  - CSNBPEX – Prohibit Export
  - CSNBPEXX – Prohibit Export Extended
  - CSNBRNG – Random Number Generate
  - CSNDRKX – Remote Key Export
  - CSNDRKA – Restrict Key Attribute
  - CSNBSKI - Secure Key Import
  - CSNBSKI2 – Secure Key Import2
  - CSNDSYX - Symmetric Key Export
  - CSNDSXD – Symmetric Key Export with Data
  - CSNDSYG – Symmetric Key Generate
  - CSNDSYI – Symmetric Key Import
  - CSNDSYI2 – Symmetric Key Import2
  - CSNDTBC – Trusted Block Create
  - CSNBT31X – TR-31 Export
  - CSNBT31I – TR-31 Import
  - CSNBT31O – TR-31 Optional Data Build
  - CSNBT31R – TR-31 Optional Data Read
  - CSNBT31P – TR-31 Parse
  - CSNBUKD – Unique Key Derive

APIs

- Protecting Data (9)
  - CSNBCTT2 – CipherText Translate2
  - CSNBDEC – Decipher
  - CSNBDCO – Decode
  - CSNBENC – Encipher
  - CSNBECO – Encode
  - CSNBSAD – Symmetric Algorithm Decipher
  - CSNBSAE – Symmetric Algorithm Encipher
  - CSNBSYD – Symmetric Key Decipher
  - CSNBSYE – Symmetric Key Encipher
- Verifying Data Integrity and Authenticating Messages (10)
  - CSNBHMG – HMAC Generate
  - CSNBHMV – HMAC Verify
  - CSNBMGN – MAC Generate
  - CSNBMGN2 – MAC Generate2
  - CSNBMVR – MAC Verify
  - CSNBMVR2 – MAC Verify2
  - CSNBMDG - MDC Generate
  - CSNBOWH – One-Way Hash Generate
  - CSNBSMG – Symmetric MAC Generate
  - CSNBSMV – Symmetric MAC Verify

APIs

- Financial Services (25)
  - CSNBAPG - Authentication Parameter Generate
  - CSNBCPE – Clear PIN Encrypt
  - CSNBPGN – Clear PIN Generate
  - CSNBCPA – Clear PIN Generate Alternate
  - CSNBCKC – CVV Key Combine
  - CSNBESC – EMV Scripting Service*
  - CSNBEAC – EMV Transaction (ARQC/ARPC) Service*
  - CSNBEVF – EMV Verification Functions*
  - CSNBEPG – Encrypted PIN Generate
  - CSNBPTR – Encrypted PIN Translate
  - CSNBPVR – Encrypted PIN Verify
  - CSNBFLD – Field Level Decipher
  - CSNBFLE – Field Level Encipher
  - CSNBFPED – FPE Decipher
  - CSNBFPEE – FPE Encipher
  - CSNBFPET – FPE Translate
  - CSNBPCU – PIN Change/Unblock
  - CSNBPFO – Recover PIN from Offset
  - CSNBSKY – Secure Messaging for Keys
  - CSNBSPN – Secure Messaging for PINs
  - CSNDSBC – SET Block Compose
  - CSNDSBD – SET Block Decompose
  - CSNBTRV – Transaction Validation
  - CSNBCSG – VISA CVV Service Generate
  - CSNBCSV – VISA CVV Service Verify
- Financial Services for DK PIN Methods (10)
  - CSNBDDPG – DK Deterministic PIN Generate
  - CSNBDMP - DK Migrate PIN
  - CSNBDPMT - DK PAN Modify in Transaction
  - CSNBDPT - DK PAN Translate
  - CSNBDPC - DK PIN Change
  - CSNBDPV - DK PIN Verify
  - CSNBDPNU - DK PRW Card Number Update
  - CSNBDPCG - DK PRW CMAC Generate
  - CSNBDRPG - DK Random PIN Generate
  - CSNBDRP - DK Regenerate PRW

APIs

- Using Digital Signatures (2)
  - CSNDDSG – Digital Signature Generate
  - CSNDDSV – Digital Signature Verify
- Managing PKA Cryptographic Keys (8)
  - CSNDPKG – PKA Key Generate
  - CSNDPKI – PKA Key Import
  - CSNDPKB – PKA Key Token Build
  - CSNDKTC – PKA Key Token Change
  - CSNDPKT – PKA Key Translate
  - CSNDPKX – PKA Public Key Extract
  - CSNDRKD – Retained Key Delete
  - CSNDRKL – Retained Key List
- Key Data Set Management (16)
  - CSNBKRC – CKDS Key Record Create
  - CSNBKRC2 – CKDS Key Record Create2
  - CSNBKRD – CKDS Key Record Delete
  - CSNBKRR – CKDS Key Record Read
  - CSNBKRR2 – CKDS Key Record Read2
  - CSNBKRW – CKDS Key Record Write
  - CSNBKRW2 – CKDS Key Record Write2
  - CSFCRC – Coordinated KDS Administration
  - CSFMPS - ICSF Multi-Purpose Service
  - CSFKDSL - Key Data Set List
  - CSFKDMR - Key Data Set Metadata Read
  - CSFKDMW - Key Data Set Metadata Write
  - CSNDKRC – PKDS Key Record Create
  - CSNDKRD – PKDS Key Record Delete
  - CSNDKRR – PKDS Key Record Read
  - CSNDKRW – PKDS Key Record Write

APIs

- Utilities (7)
  - CSNBXBC – Character/Nibble Conversion
  - CSNBXEA – Code Conversion
  - CSFIQA – ICSF Query Algorithm
  - CSFIQF – ICSF Query Facility
  - CSFIQF2 – ICSF Query Facility2
  - CSFACEE – SAF ACEE Selection
  - CSNB9ED – X9.9 Data Editing
- Trusted Key Entry Workstation Interfaces (1)
  - CSFPCI – PCI Interface Callable Service
- Using PKCS #11 Tokens and Objects (19)
  - CSFPDMK – PKCS #11 Derive Multiple Keys
  - CSFPDVK – PKCS #11 Derive Key
  - CSFPGAV – PKCS #11 Get Attribute Value
  - CSFPGKP – PKCS #11 Generate Key Pair
  - CSFPGSK – PKCS #11 Generate secret key
  - CSFPHMG – PKCS #11 Generate HMAC
  - CSFPHMV – PKCS #11 Verify HMAC
  - CSFPOWH – PKCS #11 One-way hash, sign, or verify
  - CSFPPKS – PKCS #11 Private Key Sign
  - CSFPPKV – PKCS #11 Public Key Verify
  - CSFPPRF – PKCS #11 Pseudo-random function
  - CSFPSAV – PKCS #11 Save Attribute Value
  - CSFPSKD – PKCS #11 Secret Key Decrypt
  - CSFPSKE – PKCS #11 Secret Key Encrypt
  - CSFPTRC – PKCS #11 Token Record Create
  - CSFPTRD – PKCS #11 Token Record Delete
  - CSFPTRL – PKCS #11 Token Record List
  - CSFPUWK – PKCS #11 Unwrap Key
  - CSFPWPK – PKCS #11 Wrap key

API Syntax

- All ICSF APIs start with same 4 parms
  - Return Code — INTEGER, HEX PIC 9(8) COMP
  - Reason Code — INTEGER, HEX PIC 9(8) COMP
  - Exit Data Length — INTEGER, HEX PIC 9(8) COMP
  - Exit Data — STRING, CHAR, PIC X()
- Followed by specific positional parms, unique to the API
- RULE_ARRAY
  - Array of positional parms, specific to the API
  - 8-byte keywords, left-justified, blank padded

CSNBSYE/CSNBSYD

- ICSF API for clear key encryption/decryption

CALL CSNBSYE(

return_code,
reason_code,
exit_data_length,
exit_data,
rule_array_count,
rule_array,
key_identifier_length,
key_identifier,
key_parms_length,
key_parms,
block_size,
initialization_vector_length,
initialization_vector,
chain_data_length,
chain_data,
clear_text_length,
clear_text,
cipher_text_length,
cipher_text,
optional_data_length,
optional_data)

CSNBSYE/CSNBSYD rule_array

- Rule_array(Algorithm, Processing rule, Key rule, ICV selection)
  - Algorithm: AES, DES
  - Processing rule: CBC, CBC-CS, CFB, CFB-LCB, CTR, CUSP, ECB, GCM, GCM-LG, IPS, OFB, PKCS-PAD, X9.23)
  - Key Rule: KEY-CLR, KEYIDENT
  - ICV Selection: INITIAL, CONTINUE, FINAL, ONLY
- Positional parameters, required/optional
- 8-character, padded

CSNBSYE Example - variables

```
api_rc                  = 'FFFFFFFF'x ;
api_rs                  = 'FFFFFFF0'x ;
api_exit_data_length    = '00000000'x ;
api_exit_data           = '' ;
sye_rule_array_cnt      = '00000004'x ;
sye_rule_array          = 'AES     CBC     KEYIDENTINITIAL ' ;
sye_keyid_length        = '00000040'x ;                  /* decimal 64 */
sye_keyid               = 'MYKEYLBL';
sye_keyparms_len        = '00000000'x ;  /* Ignored with CBC mode */
sye_keyparms            = '' ;           /* Ignored with CBC mode */
```

CSNBSYE Example - variables

```
sye_blksize          = '00000010'x ;       /*  Decimal 16 for AES */
sye_icv_len          = '00000010'x ;          /* same as blksize     */
sye_icv              = '00112233445566778899AABBCCDDEEFF'x;
sye_chaindat_len     = '00000020'x ;       /* must be 32 for AES  */
sye_chaindata        = copies('00'x,32) ;
sye_clear_text_len   = '00000020'x ;
sye_clear_text       = 'Our 32 char message with padding' ;
sye_cipher_text_len  = '00000020'x ;
sye_cipher_text      = copies('40'x,32) ;       /* our output field */
sye_optdata_len      = '00000000'x ; /* ignored except for GCM */
sye_optdata          = '' ;
```

CSNBSYE - Invocation

```
address linkpgm 'CSNBSYE' ,
                'api_rc',
                'api_rs',
                'api_exit_data_length' ,
                'api_exit_data' ,
                'sye_rule_array_cnt',
                'sye_rule_array',
                'sye_keyid_length',
                'sye_keyid',
                'sye_keyparms_len',
                'sye_keyparms',
                'sye_blksize',
                 'sye_icv_len',
                 'sye_icv',
                 'sye_chaindat_len',
                 'sye_chaindata',
                 'sye_clear_text_len',
                 'sye_clear_text',
                 'sye_cipher_text_len',
                 'sye_cipher_text',
                 'sye_optdata_len',
                 'sye_optdata'
```

CSNBSYE - Results

```
/* check the return and reason codes */
if (API_rc = '00000000'x & API_rs = '00000000'x)
   then do;
       say 'SYE OK, rc=' c2x(API_rc) 'rs =' c2x(API_rs) ;
      end ;
   else do ;
      say 'SYE failed, rc =' c2x(API_rc) 'rs =' c2x(API_rs) ;
      exit;
   end ;

say 'sye_clear_text(ch) = 'sye_clear_text';' ;
say 'sye_cipher_text(ch) = 'sye_cipher_text';' ;
say 'sye_cipher_text(hex) = 'c2x(sye_cipher_text)';' ;
```

ENCIPHER/DECIPHER API

- CALL CSNBENC (
  - return code,
  - reason code,
  - exit data length,
  - exit data,
  - key identifier,
  - text length,
  - clear text,
  - initialization vector,
  - rule array count,
  - rule array,
  - pad character,
  - chaining vector,
  - cipher text)

- CALL CSNBDEC (
  - return code,
  - reason code,
  - exit data length,
  - exit data,
  - key identifier,
  - text length,
  - clear text,
  - initialization vector,
  - rule array count,
  - rule array,
  - chaining vector,
  - cipher text)

See the ICSF Programmer's Guide for all of the APIs including the parms for each API

Rule Array (for CSNBENC/CSNBDEC)

- Position 1 (required)
  - CBC
  - CUSP
  - IPS
  - X9.23
  - 4700-PAD

  Has to do with blocking and chaining

- DES Algorithm (optional)
  - DES
  - TOKEN

  Can override some of the CCA bit settings related to how keys are used (encrypting data vs other keys)

- ICV Selection (optional)
  - CONTINUE
  - INITIAL

  INITIAL says to use initialization vector, not the chaining vector

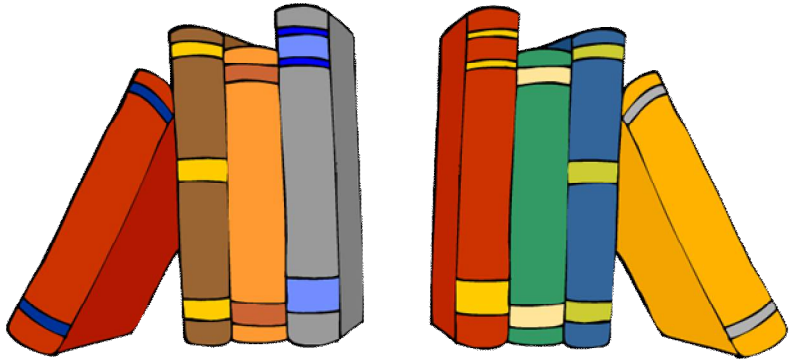
ICSF SAF Security – CSFSERV Class

- RACF
  RDEFINE CSFSERV service_name UACC(NONE)
  PERMIT service_name CLASS(CSFSERV) ID(userid) ACCESS(READ)

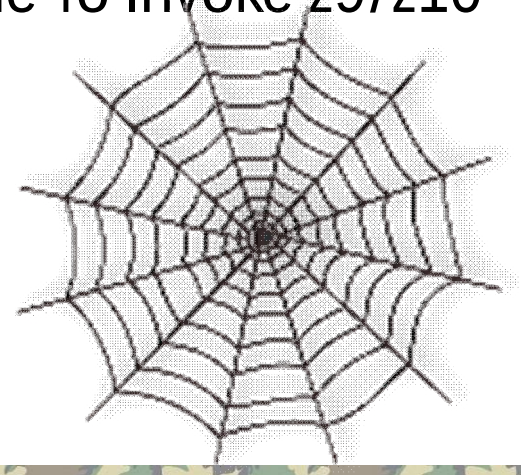
- CA TopSecret®for z/OS
  TSS ADD(owner) CSFSERV(service_name)
  TSS PERMIT(userid/profile) CSFSERV(service_name) ACCESS(READ)

- CA ACF2™ for z/OS
  SET RESOURCE(SAF)
  COMPILE
  $KEY(service) TYPE(SAF)
  UID(userid) SERVICE(READ) ALLOW
  STORE

References



- IBM Pubs
  - SA22-7832 Principles of Operations (-10 z13)
  - SC14-7508 ICSF Application Programmer's Guide (z/OS V2)
  - SA22-7522 ICSF Application Programmer's Guide (z/OS V1)
- Techdocs – www.ibm.com/support/techdocs
  - PRS2669 – CPACFZ9S – How to Use The z9/z10 CPACF Crypto Functions
  - PRS822 – CALCPACF: Callable z/OS Routine To Invoke z9/z10 CPACF Crypto Functions
  - Or search on 'crypto'



Questions