An Introduction to z/OS Cryptographic Key Management

Eysha S. Powers
eysha@us.ibm.com
Enterprise Cryptography

What is cryptography?

Cryptography is defined as the practice and study of techniques for secure communication in the presence of third parties (i.e. adversaries).

- Confidentiality – Preventing the disclosure of information to unauthorized individuals.
    - **Encrypt:** Convert clear text to cipher text
    - **Decrypt:** Convert cipher text to clear text
- Integrity – Maintaining and assuring the accuracy and consistency of data.
    - **Hash:** Translate clear text to a fixed length hash value

      Example (32-byte hash):
      1025 4AD0 04D2 C7D5 77EA ADA0 E4C8 B76F A290 2F7C D03B F03E B527 A045 E200 238F

    - **Sign:** Hash the clear text and encrypt the hash with a private key
    - **Verify:** Hash the clear text then decrypt the sender's hash using the sender's public key and compare the hash values
- Authentication – Verifying the identity of a party.
- Non-repudiation – Assuring that a party cannot deny that they created a message.

What are cryptographic keys?

Symmetric keys are simply a sequence of bits of a precise length (i.e. key size) intended for use in a cryptographic operation.

- DES = 56 bits (i.e. 8 bytes)
- TDES = 56, 112, or 168 bits (i.e. 8, 16 or 24 bytes)
- AES = 128, 192, or 256 bits (i.e. 16, 24 or 32 bytes)

Where do symmetric key bytes come from?

- Random number generators
  - True random number generation requires:
    - An entropy source of randomness **to**
    - Produce true random bytes
  - Pseudo Random number generation requires:
    - An entropy source of randomness **PLUS**
    - A deterministic mathematical algorithm **to**
    - Produce pseudo random bytes

Tip: Invoke /dev/random for random number generation from the z/OS Unix System Services shell.

Asymmetric key pairs are generated using complex math operations. They typically rely on trap door functions which are easy to compute in one direction but difficult to compute in the opposite direction.

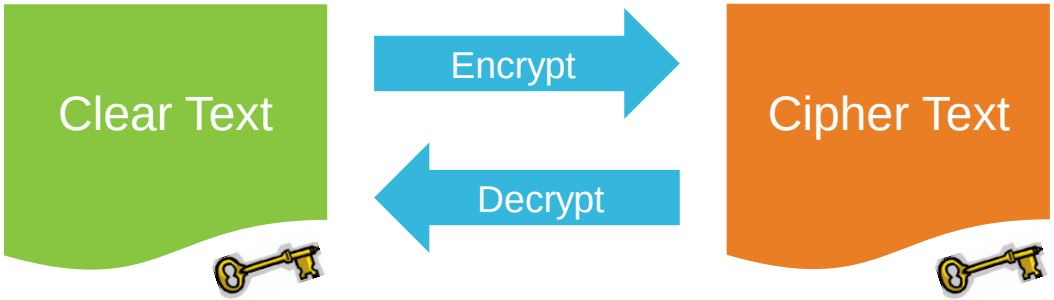
- RSA = 1024 – 4096 bits
- EC BrainPool = 160, 192, 224, 256, 320, 384, or 512
- EC Prime = 192, 224, 256, 384 or 521

Why does the key length matter?

- Short key lengths, specifically for symmetric keys, can be brute force attacked, especially with today's computing speeds
  - The NIST standards body recommends symmetric keys of 24 bytes or larger.
- Long key lengths, specifically for asymmetric keys, take much more time to generate
  - ECC key pairs offer stronger encryption than RSA with smaller key sizes

How are key values used for encryption and decryption?

- Provide a ***key value*** and ***clear text*** to a cryptography algorithm to produce ***cipher text*** (i.e. encryption)
- Provide a ***key value*** and ***cipher text*** to a cryptography algorithm to produce ***clear text*** (i.e. decryption)



For symmetric encryption, the encryption key and decryption key are the same!

The Anatomy of a Fixed-Length Key Token

Internal AES fixed-length CCA key token (64 bytes)

```
01000000400C0D1F506FF13A5FFC26AE682510712FCF2327584F83E8289B966
F407117FD08A16825BDEBD86BE4F2647000000000000000000100002071548319
```

Bytes	Description
0	X'01' flag indicating an internal key token
1 - 3	X'000000' for ICSF
4	Key token version number (X'04')
5	Reserved – must be set to X'00'
6	Flag byte
7	1-byte Longitudinal Redundancy Check (LRC) checksum of a clear key value
8 - 15	Master key verification pattern (MKVP)
16 - 47	Key value, if present
48 - 55	8-byte control vector (For a clear AES key token this value will be hex zeroes.)
56 - 57	2-byte integer specifying the length in bits of the clear key value
58 - 59	2-byte integer specifying the length in bytes of the encrypted key value.
60 - 63	Token validation value (TVV)

AES = Advanced Encryption Standard

CCA = Common Cryptographic Architecture

AES key values may be 16, 24 or 32 bytes.

See the Cryptographic Services Integrated Cryptographic Service Facility Application Programmer's Guide for additional details

What happens when a key value is exposed or compromised?

If the key value was in the clear

- The key value can be used to decrypt sensitive data

If the key value was encrypted

- The key value cannot be used to decrypt sensitive data without the associated key encrypting key (KEK)

The Anatomy of a Fixed-Length Key Token

Internal AES fixed-length CCA key token (64 bytes)

Bytes	Description
0	X'01' flag indicating an internal key token
1 - 3	X'000000' for ICSF
4	Key token version number (X'04')
5	Reserved – must be set to X'00'
6	Flag byte
7	1-byte Longitudinal Redundancy Check (LRC) checksum of a clear key value
8 - 15	Master key verification pattern (MKVP)
16 - 47	Key value, if present
48 - 55	8-byte control vector (For a clear AES key token this value will be hex zeroes.)
56 - 57	2-byte integer specifying the length in bits of the clear key value
58 - 59	2-byte integer specifying the length in bytes of the encrypted key value.
60 - 63	Token validation value (TVV)

AES = Advanced Encryption Standard

CCA = Common Cryptographic Architecture

AES key values may be 16, 24 or 32 bytes.

See the Cryptographic Services Integrated Cryptographic Service Facility Application Programmer's Guide for additional details

What are key encrypting keys (KEKs)?

KEKs are keys that protect (e.g. encrypt, wrap) other keys

Master Keys

Master keys are used only to encipher and decipher keys.

Master keys are stored in secure, tamper responding hardware.

Master key encrypted keys are considered secure keys.

Master keys should be changed periodically.

All master keys are optional. Secure keys are only supported when their associated master key is active.

Operational Keys

Operational keys are used in various cryptographic operations (e.g. encryption).

Operational keys may be stored in a key store (e.g. data set, file, database) or returned back to the caller.

Operational keys may be clear, secure or protected.

Symmetric KEKs

Encrypt symmetric keys with another symmetric key.

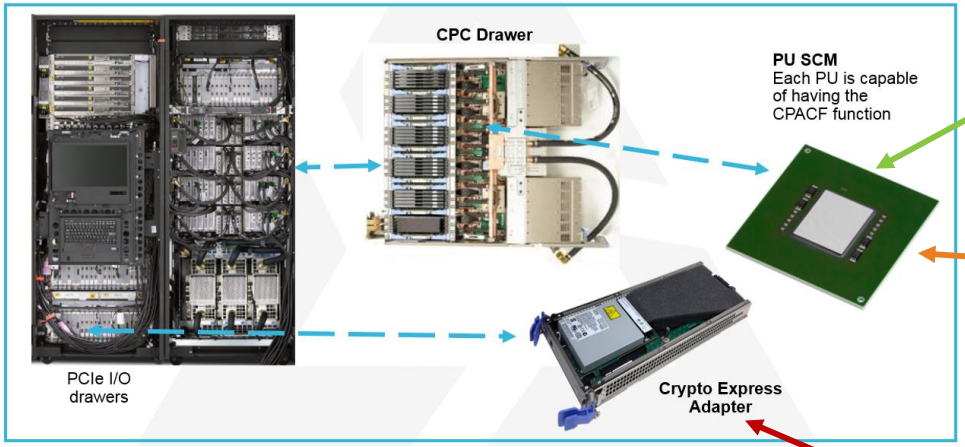
Asymmetric KEKs

Encrypt symmetric keys with RSA public keys

Use ECC key pairs to derive a symmetric key. Use the derived symmetric key to encrypt another symmetric key.

Understanding Clear, Secure and Protected Keys

Secure keys have key values that are encrypted by a Master Key on a tamper-responding CryptoExpress adapter.



High speed

Clear Key

Key values are not encrypted. Crypto operations may be performed in CPACF or on a Crypto Express adapter

Speed & Security Hybrid

Protected Key

Key values are encrypted under a CPACF wrapping key. Crypto operations are performed only using CPACF

High security

Secure Key

Key values are encrypted under a Master Key. Crypto operations are performed only on a Crypto Express adapter

Only protected keys created from secure keys should be used for Pervasive Encryption.

Note: With z/OS data set encryption, protected keys are implicitly created from secure keys.

How do you generate, maintain and manage Master Keys?

- Using the Trusted Key Entry (TKE) Workstation
  - Applicable for initialization of ICSF Key Data Sets (i.e. key stores) and Crypto Express adapters
  - Applicable for master key change operations
  - Required for EP11 Master Key management & PCI-HSM Master Key management
  - Separate, priced product

Trusted Key Entry (TKE) Workstation

Smart Cards

Smart Card Readers

- Using the ICSF Master Key Entry Panels
  - Applicable for initialization of ICSF Key Data Sets (i.e. key stores) and Crypto Express adapters
  - Applicable for master key change operations
  - Included with z/OS and ICSF

- Using the Pass Phrase Initialization (PPINIT) Panel
  - Applicable for initialization of ICSF Key Data Sets (i.e. key stores) and Crypto Express adapters
  - **NOT** applicable for master key change operations
  - Included with z/OS and ICSF

Special Considerations for Master Keys

- Master Keys are high value keys that must be protected.
  - Loading Master Keys on a panel means that the key is viewable to passersby!
  - The most secure way to load a Master Key is to use the TKE Workstation with smart cards.
    - The P11 Master Key may ONLY be loaded using a TKE Workstation.
- If you plan to use the PPINIT or the Master Key Entry panels to manage Master Keys, consider how you would save the key material for future re-entry (e.g. new Crypto Express adapter, disaster recovery).
- For disaster recovery, the same Master Keys must be loaded onto the backup system.

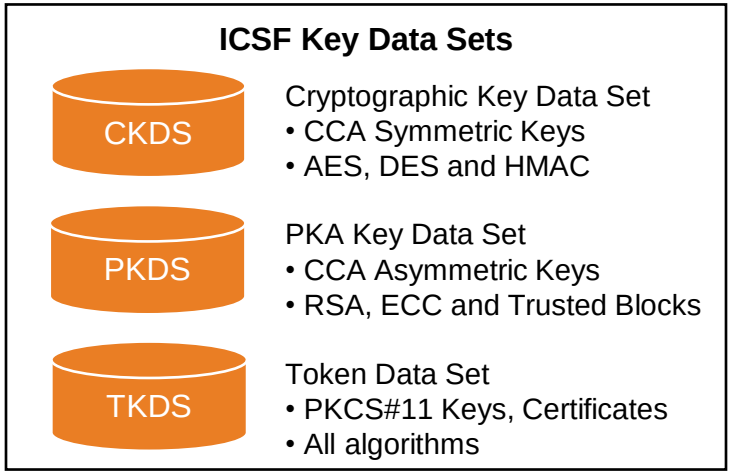
Option	Details	Pros	Cons
Print Screen	Use a Print Screen key or tool to capture the screen	Sensitive material can be immediately printed and stored in envelopes in a locked safe. No need to save on a local machine or USB stick.	Cannot use copy / paste to re-enter key material
Removable Storage Media	Copy and paste key material to a text file that is saved on a secure storage device (e.g. USB stick).	Easy to copy / paste the key material to the panels for re-entry.	The key material is only as secure as the storage media.
Other Ideas?			

How does ICSF generate, maintain and manage operational keys?

- ICSF provides callable services and utilities to generate and store operational keys into ICSF Key Data Sets (KDS) and/or return the keys to the caller
- Each KDS is a VSAM data set for persistent objects (e.g. keys, certificates) with programming interfaces for object management.
- Each record in the KDS contains the object and other information about that object.

ICSF uses keys in cryptographic functions to

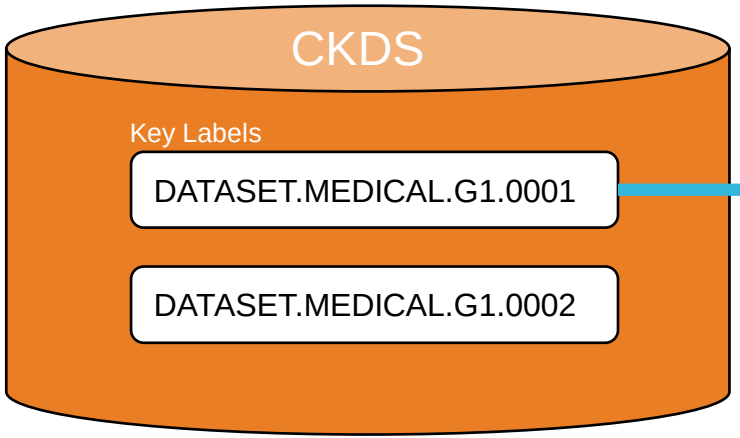
- Protect data
- Protect other keys
- Verify that messages were not altered
- Generate, protect and verify PINs
- Distribute keys
- Generate and verify signatures



Anatomy of a Key Record

Common Record Format CKDS LRECL=2048

Key lookup is performed using a key label (CKDS, PKDS) or key handle (TKDS).



The CKDS key material can be an AES, DES or HMAC key token

Offset	Number of Bytes	Field Name
0	72	Key label or handle
72	8	Reserved
80	1	Version
81	1	KDS type (CKDS, PKDS, TKDS)
82	2	Flags
84	4	Record length
88	8	Creation date
96	8	Creation time
104	8	Last update date
112	8	Last update time
120	4	Key material length
124	4	Key material offset
128	4	Metadata length
132	4	Metadata offset
136	4	Reserved

Note: Common Record Format was introduced in ICSF HCR77A1 for z/OS v1r13 and later

Understanding Key Labels

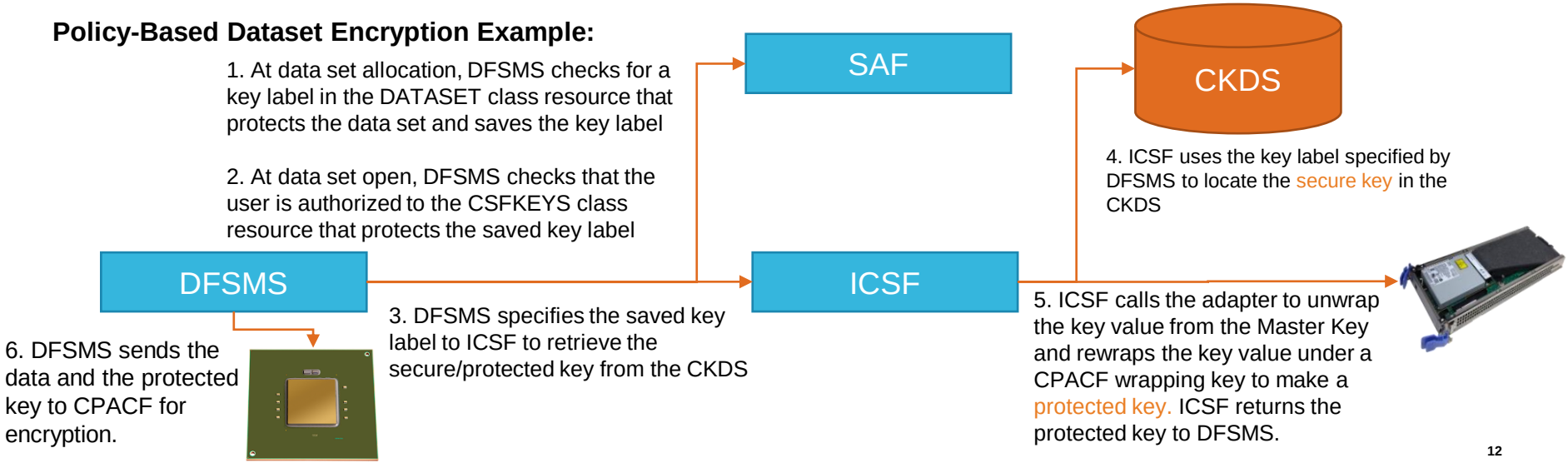
Every record in the CKDS has an associated key label.

When user applications or z/OS components invoke ICSF callable services (i.e. APIs), the application can specify a key label as a parameter to identify the key for the callable service to use.

System Authorization Facility (SAF) policies control which users can use which keys (and callable services).

- The CSFKEYS class controls access to cryptographic keys in the ICSF CKDS and PKDS and enables/disables the use of protected key.
- The CSFSERV class controls access to ICSF callable services and ICSF TSO panel utilities.

Policy-Based Dataset Encryption Example:



Key Label Naming Conventions & Access Control

The CSFKEYS SAF class controls access to cryptographic keys in the ICSF Key Data Sets (CKDS and PKDS) and enables/disables the use of protected keys.

With RACF-based SAF protection, CSFKEYS resources can be defined as discrete or generic (i.e. wildcard) profiles. As a result, *KDS key label naming conventions are important*.

A key label can consist of up to 64 characters. The first character must be alphabetic or a national character (#, $, @). The remaining characters can be alphanumeric, a national character (#, $, @), or a period (.).

Naming considerations:

- the LPAR associated with the key
- the type of data being encrypted
- the owner associated with the key
- the date the key was created
- the application intended to use the key
- The generic profile to protect the key
- A sequence number for the key

Policy-Based Dataset Encryption Example:

Key Label:

DATASET.<dataset_resource>.ENCRKEY.<seqno>

CSFKEYS Profile:

RDEFINE CSFKEYS DATASET. <dataset_resource>.ENCRKEY.* UACC(NONE)

Note: <dataset_resource> would be replaced with the DATASET resource and <seqno> would be replaced with a sequence number.

Additional Metadata

The metadata section of the Common Record Format KDS can be used to store up to 500 bytes of custom installation data.

The Key Dataset Metadata Write (CSFKDMW) and Key Dataset Metadata Read (CSFKDMR) callable services can be invoked to read and write metadata.

Example Metadata:
- Key owner's name
- Key owner's email address
- Reference to data being encrypted (e.g. dataset name)
- Comments about the key and/or data encrypted by the key

Samples for using CSFKDMW and CSFKDMR are available on the IBM Crypto Education Community… https://ibm.biz/BdjcFx

Offset	Number of Bytes	Field Name
0	72	Key label or handle
72	8	Reserved
80	1	Version
81	1	KDS type (CKDS, PKDS, TKDS)
82	2	Flags
84	4	Record length
88	8	Creation date
96	8	Creation time
104	8	Last update date
112	8	Last update time
120	4	Key material length
124	4	Key material offset
128	4	Metadata length
132	4	Metadata offset
136	4	Reserved

Metadata support requires ICSF HCR77B0 or later and a Common Record Format KDS

What is the relationship between a key record, a key token and a key value?

Key Record

Offset	Number of Bytes	Field Name
0	72	Key label or handle
72	8	Reserved
80	1	Version
81	1	KDS type (CKDS, PKDS, TKDS)
82	2	Flags
84	4	Record length
88	8	Creation date
96	8	Creation time
104	8	Last update date
112	8	Last update time
120	4	Key material length
124	4	Key material offset
128	4	Metadata length
132	4	Metadata offset
136	4	Reserved

Key Value

Key Token

The Anatomy of a Fixed-Length Key Token

Internal AES fixed-length CCA key token (64 bytes)

Bytes	Description
0	X'01' flag indicating an internal key token
1 - 3	X'000000' for ICSF
4	Key token version number (X'04')
5	Reserved – must be set to X'00'
6	Flag byte
7	1-byte Longitudinal Redundancy Check (LRC) checksum of a clear key value
8 - 15	Master key verification pattern (MKVP)
16 - 47	Key value, if present
48 - 55	8-byte control vector (For a clear AES key token this value will be hex zeroes.)
56 - 57	2-byte integer specifying the length in bits of the clear key value
58 - 59	2-byte integer specifying the length in bytes of the encrypted key value.
60 - 63	Token validation value (TVV)

AES = Advanced Encryption Standard

CCA = Common Cryptographic Architecture

AES key values may be 16, 24 or 32 bytes.

See the Cryptographic Services Integrated Cryptographic Service Facility Application Programmer's Guide for additional details

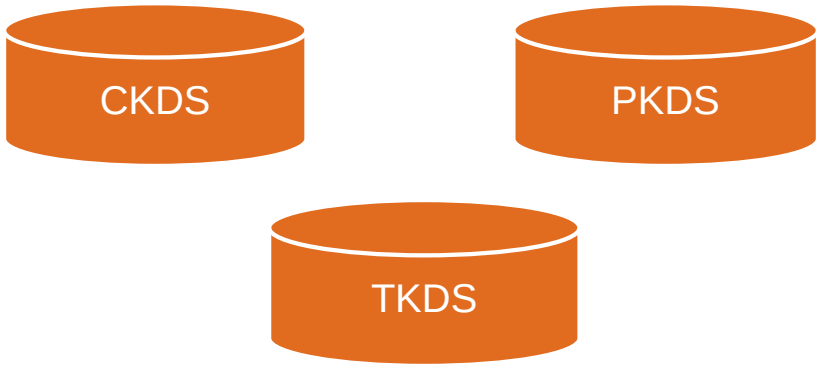
The key record contains a key token which contains a key value.

How do you create a Common Record Format KDS?

Step 1:

Allocate new Key Data Sets.

- **CKDS:** SYS1.SAMPLIB(CSFCKD3)
- **PKDS:** No change to allocation process
- **TKDS:** SYS1.SAMPLIB(CSFTKD2)



See the next slide for CKDS allocation considerations.

Step 2:

A: If there are no existing keys to convert then

- Initialize new Key Data Sets using the ICSF panels (all KDS types) or JCL job (TKDS)

B: If there are existing keys to convert to the new format

- Run the KDS Conversion utility from the ICSF KDS Management panels (for each KDS type to be converted)

CKDS Allocation Considerations

The amount of primary space required for the CKDS depends on the number of keys the dataset will initially contain.

Primary Space = initial key count * record size

For example:

Initial load of 10K keys, all fixed length tokens.

Primary Space = 10K * 744 = approx. 7.3 MB

The maximum record size of a DATA key = 140-byte header + 40-byte metadata section + 64-byte key token + 500 bytes of metadata = 744 bytes

The amount of secondary space depends on how many keys will be added.

Secondary Space = future key count * record size

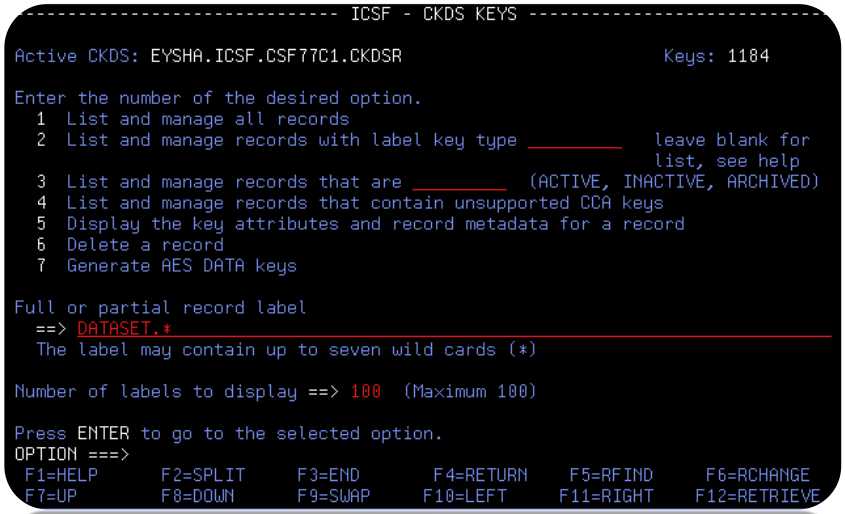
For example, 83K keys added every year for 10 years = 830K keys

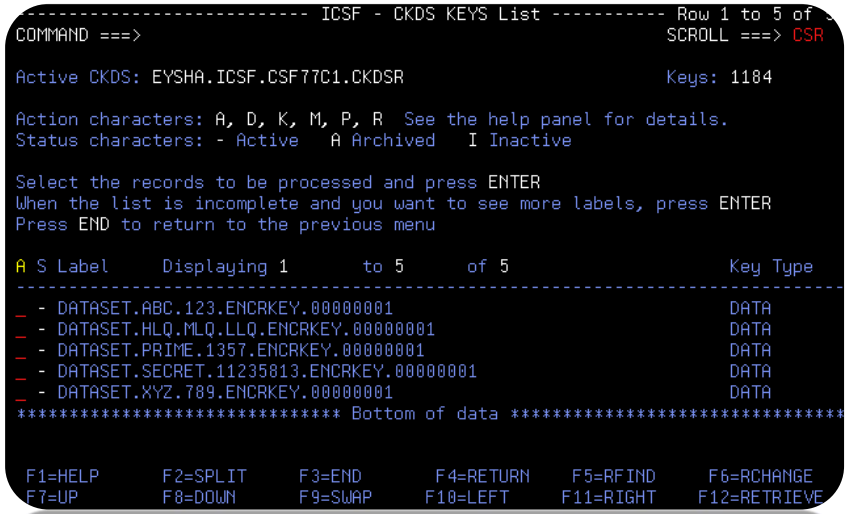
Secondary Space = 830K * 744 = approx. 603 MB

How do you view the contents of a Key Data Set?

With HCR77C1, ICSF supports a CKDS Browser (ICSF Panel Option 5.5).

Note: Alternative methods include IDCAMS REPRO, PKCS #11 Token (TKDS) Browser and the Key Dataset List (CSFKDSL) callable service.



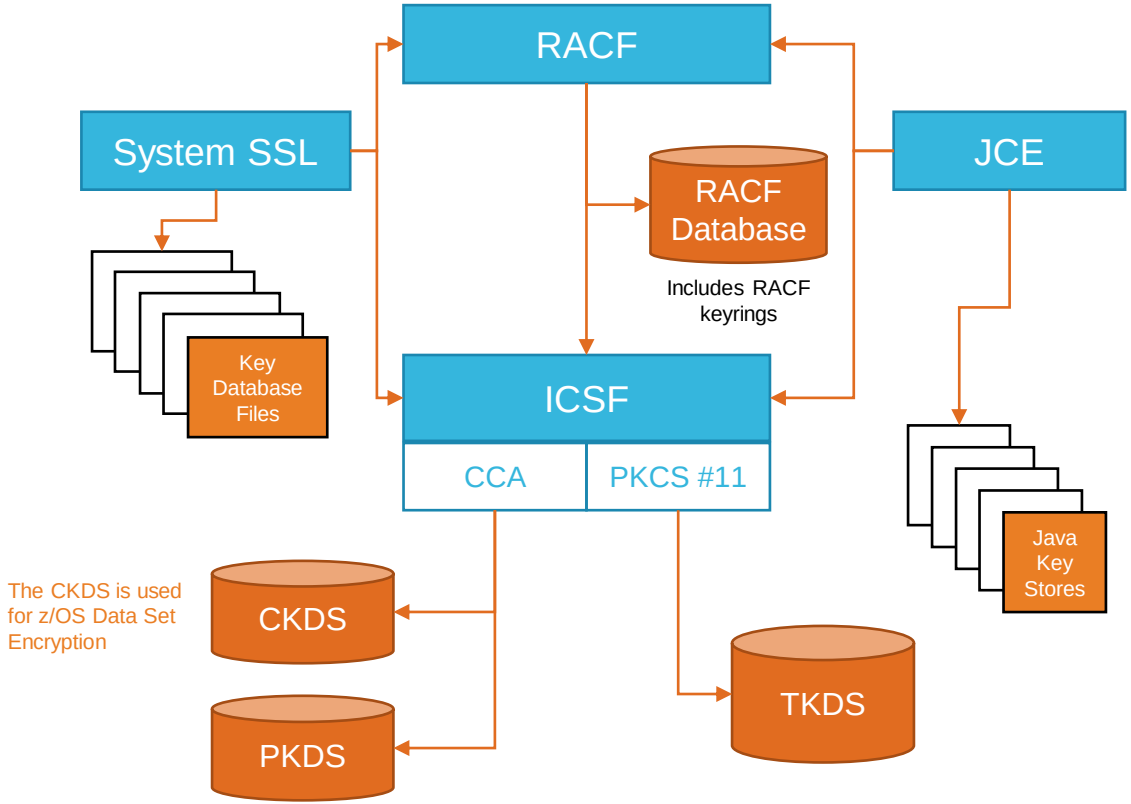


Additional z/OS Key Stores

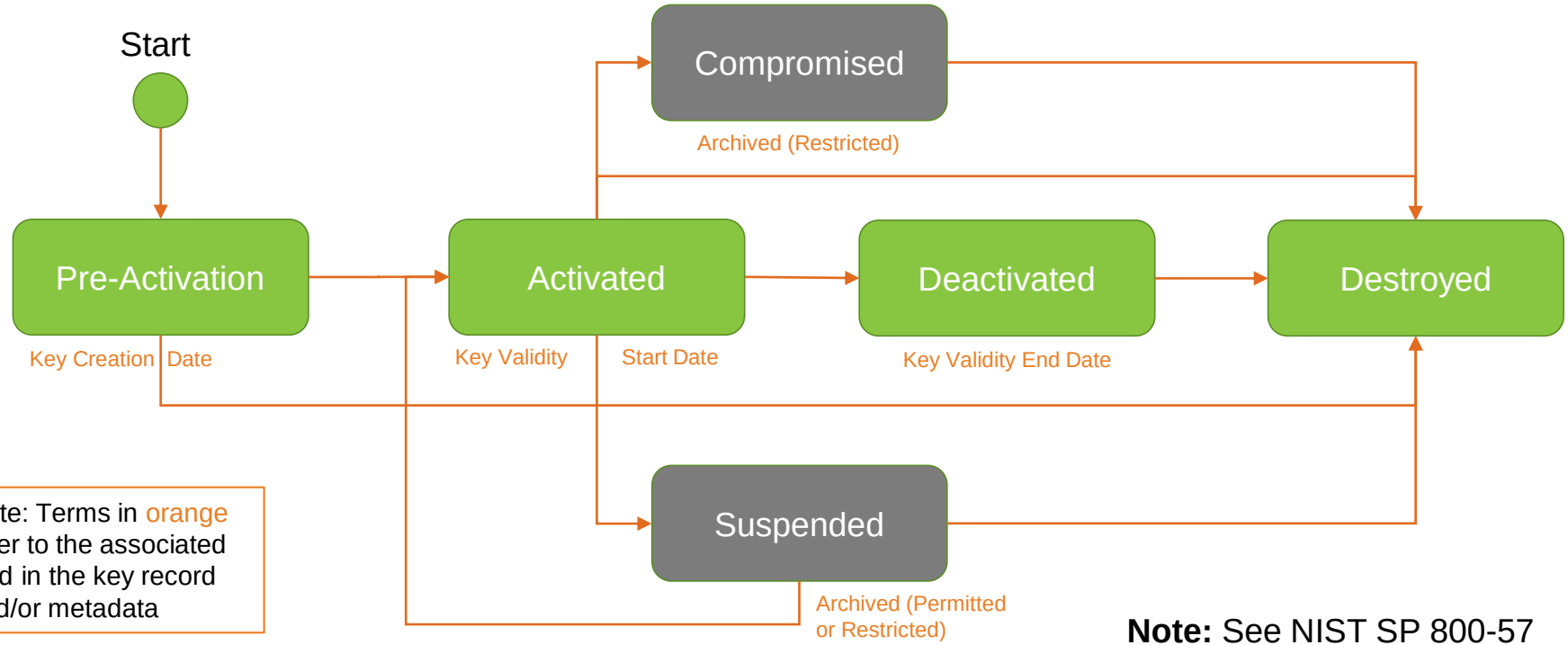
- **RACF** provides the RACDCERT GENCERT command to generate and store keys into the RACF database and ICSF Key Data Sets (PKDS and TKDS). RACF also provides the RACDCERT CONNECT command to add certificates to RACF Keyrings.

- **SystemSSL** provides the gskkyman utility to generate and store certificates into key database files. SystemSSL can also read from RACF Keyrings and generate and store certificates into PKCS#11 Tokens (TKDS).

- **JCE** provides APIs and utilities to generate and store keys and certificates into ICSF Key Data Sets, RACF Keyrings, and Java Key Stores.



Key Life Cycle (Simple View)



Note: Terms in orange refer to the associated field in the key record and/or metadata

Note: See NIST SP 800-57 for additional state transitions

Locating Key Life Cycle Metadata in KDS Records

Tag	Meaning
X'0001'	Variable metadata block
X'0002'	Record create date
X'0003'	Record update date
X'0004'	Key material validity start date
X'0005'	Key material validity end date
X'0006'	Last reference date (YYYYMMDD)
X'0007'	Last reference date (first 8 bytes of the value returned by store clock extended instruction)
X'0008'	Record archive date
X'0009'	Record archive flag
X'000A'	Record prohibit archive flag
X'000B'	Record recall date

Tag	Meaning
X'0001'	Installation user data
X'0002'	Service for reference
X'0003'	Record archive date
X'0004'	Record recall date
X'0005'	Key fingerprint
X'0006'	Retained key information
X'8000' - X'FFFF'	Installation metadata

Remember…

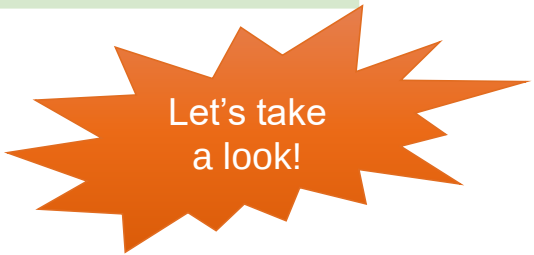
Metadata support requires ICSF HCR77B0 or later and a Common Record Format Key Data Set

Key Dataset Metadata Write (CSFKDMW) and Key Dataset Metadata Read (CSFKDMR) callable services can be invoked to read and write metadata.

Is there a way to audit key life cycle transitions?

Key life cycle auditing must be explicitly enabled in the ICSF Installation Options Data Set (IODS) or the SETICSF OPT operator commands.

ICSF IODS Option	SMF Record Type
AUDITKEYLIFECKDS(TOKEN(YES),LABEL(YES))	Type 82 Subtype 40
AUDITKEYLIFEPKDS(TOKEN(YES),LABEL(YES))	Type 82 Subtype 41
AUDITKEYLIFETKDS(TOKENOBJ(YES),SESSIONOBJ(YES))	Type 82 Subtype 42



SMF Record Type 82 Subtype 40

CCA Symmetric Key Lifecycle Event

Tag	Name	Description
X’0100’	KEY_EVENT	Key event.
X’0101’	KDS_LABEL	The label in the KDS
X’0102’	KDS_DSNAME	The data set name of the KDS associated with the event.
X’0103’	KEY_NAME	The key name from the token. Applies to variable-length CCA tokens only.
X’0105’	KEY_FPRINT	One or more key fingerprints.
X’0106’	SERVICE	The service associated with the event.
X’0108’	TOK_FMT	The format of the token.
X’0109’	KEY_SEC	Key security.
X’010A’	KEY_ALG	Key algorithm.
X’010B’	KEY_TYPE	Key type. Applies to variable-length CCA tokens only.
X’010C’	KEY_CV	Key control vector. Applies to fixed-length DES CCA tokens only.
X’010D’	KEY_USAGE_CKDS	Key usage fields. Applies to variable-length CCA tokens only.
X’010E’	KEY_LEN	The length of the key (in bits). Applies to fixed-length CCA tokens only.
X’010F’	KEY_CP	Key crypto period.
X’0118’	KEY_TIV	A key token identification value. Applies to fixed-length CCA tokens only.
X’0119’	KEY_COMP_TAG	The key is compliant tagged. Applies to fixed-length CCA tokens only.

CCA Symmetric Key Lifecycle Events

X'10'	Key token added to KDS.
X'11'	Key token updated in KDS.
X'12'	Key token deleted from KDS.
X'13'	Key token archived.
X'14'	Key token restored.
X'15'	Key token metadata changed.
X'17'	Key token pre-activated.
X'18'	Key token activated.
X'19'	Key token deactivated.
X'1B'	Key token exported.
X'20'	Key token generated.
X'21'	Key token imported.

SMF Record Type 82 Subtype 40

CCA Symmetric Key Lifecycle Event

```
Subtype=0028 CCA Symmetric Key Lifecycle Event
Written for lifecycle events related to symmetric CCA tokens
25 Jul 2017 20:12:50.09
   TME... 006F09D1 DTE... 0117206F SID... SP21    SSI... 00000000 STY... 0028
   KEV... Key Generated
   SRV... CSFKGN
   TOKFMT Fixed
   KALG.. AES
   KSEC.. Wrapped by MK
   KLEN.. 256
   TIV... '60B9EB9F'x
   KFP... 010105AE36E9
          ENCZ.. 'AE36E9'x
```

```
ICSF Server Identity...
   USRI.. SYSTASK
   GRPN.. SYS1
   JBN... CSFEPC1
   RST... 13:39:45.82
   RSD... 25 Jul 2017
   SUID.. 4040404040404040
```

```
End User Identity...
   USRI.. EYSHA
   GRPN.. SYS1
   TRM... LOCALC11
   JBN... EYSHA
   RST... 18:48:59.40
   RSD... 25 Jul 2017
   SUID.. 4040404040404040
```

How do you control key usage?

System Authorization Facility (SAF) Policies

- The CSFKEYS class controls access to cryptographic keys in the ICSF Key Data Sets (CKDS and PKDS) and enables/disables the use of protected keys.
  - The SYMCPACFWRAP field of the ICSF segment enables you to specify whether ICSF can rewrap the encrypted key using the CPACF wrapping key.
  - The SYMCPACFRET field of the ICSF segment enables you to specify whether ICSF can return the protected-key form of the CCA token to a caller.
- The CSF.* resources in the XFACILIT class define rules for the user of encrypted key tokens that are stored in the CKDS and PKDS.

Control Vectors

A control vector ensures that an operational key can only be used in cryptographic operations for which it is intended.

> For example, the control vector for a DATA key ensures that such a key can be used only in the data encryption and decryption functions.

Control vectors are only supported for fixed-length DES CCA key tokens.

Fixed-length AES CCA key tokens have a zeroed control vector. These keys can only be created as DATA keys to be used for data encryption and decryption. There are no variants.

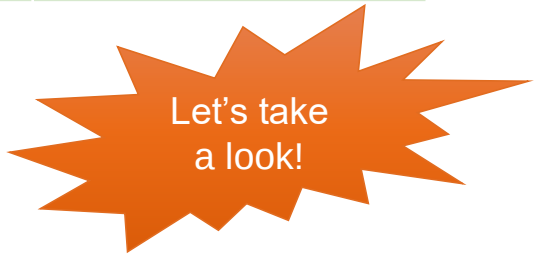
Note: Variable-length symmetric key tokens provide key-management fields (*kmf*) and key-usage fields (*kuf*) to control key usage.

Is there a way to audit key usage?

Key usage auditing must be explicitly enabled in the ICSF Installation Options Data Set (IODS) or using the SETICSF OPT operator commands.

ICSF IODS Option	SMF Record Type
AUDITKEYUSGCKDS(TOKEN(YES),LABEL(YES),INTERVAL(*n*))	Type 82 Subtype 44
AUDITKEYUSGPKDS(TOKEN(YES),LABEL(YES),INTERVAL(*n*))	Type 82 Subtype 45
AUDITPKCS11USG(TOKENOBJ(YES),SESSIONOBJ(YES),NOKEY(YES), INTERVAL(*n*))	Type 82 Subtype 46 & Type 82 Subtype 47

Note: The INTERVAL in which the key usage data is aggregated can be from 1 to 24 hours in the Installation Options Data Set. However, it can be from 1 second to 24 hours using the SETICSF OPT operator command.



SMF Record Type 82 Subtype 44

CCA Symmetric Key Usage Event

Tag	Name	Description
X'0101'	KDS_LABEL	The label in the KDS
X'0103'	KEY_NAME	The key name from the token. Applies to variable-length CCA tokens only.
X'0105'	KEY_FPRINT	One or more key fingerprints.
X'0106'	SERVICE	The service associated with the event.
X'0108'	TOK_FMT	The format of the token.
X'0109'	KEY_SEC	Key security.
X'010A'	KEY_ALG	Key algorithm.
X'010B'	KEY_TYPE	Key type. Applies to variable-length CCA tokens only.
X'010C'	KEY_CV	Key control vector. Applies to fixed-length DES CCA tokens only.
X'010D'	KEY_USAGE_CKDS	Key usage fields. Applies to variable-length CCA tokens only.
X'010E'	KEY_LEN	The length of the key (in bits). Applies to fixed-length CCA tokens only.
X'0113'	START_TOD	Start time of the interval in STCKE format.
X'0114'	END_TOD	End time of the interval in STCKE format.
X'0115'	USG_COUNT	Number of usages accounted for in this record
X'0116'	KEY_OLD	The key is internal, but not wrapped under the current master key.
X'0118'	KEY_TIV	A key token identification value. Applies to fixed-length CCA tokens only.
X'0119'	KEY_COMP_TAG	The key is compliant tagged. Applies to fixed-length CCA tokens only.

Remember that fixed-length AES CCA tokens always have a zeroed control vector so neither of these fields apply.

SMF Record Type 82 Subtype 44

CCA Symmetric Key Usage Event

```
Subtype=002C CCA Symmetric Key Usage Event
Written for usage events related to symmetric CCA tokens
25 Jul 2017 21:44:22.97
   TME... 00776B79 DTE... 0117206F SID... SP21    SSI... 00000000 STY... 002C
   STOD.. 07/26/2017 01:44:21.974598
   ETOD.. 07/26/2017 01:44:22.974680
   SRV... CSFKRR2
   USGC.. 2
   LBL... DATASET.EYSHA.ICSF.ENCRYPT.ME.ENCRKEY.00000001                  DATA
   TOKFMT Fixed
   KALG.. AES
   KSEC.. Wrapped by MK
   KLEN.. 256
   TIV... 'AFBEB90C'x
   KFP... 010105502B47
          ENCZ.. '502B47'x
End User Identity...
   USRI.. DATAOWN
```

An ICSF audit section which is supported with SMF Record Type 82 Subtype 40 and higher may contain additional audit information. For example, the end user RACF user id associated that used the key.

SMF Record Type 82 Subtype 28

High Performance Encrypted Key

Name	Description
SMF82HPSK_FLAGS	High performance encrypted key flags Bit 0: Rewrapping not permitted for this symmetric key Bit 1: Rewrapping was permitted for this symmetric key. Bit 2: The list of labels is incomplete. Bit 3: The key identifier was supplied as a key token, not as a label in the CKDS.
SMF82HPSK_FUNCTION	Name of the service that issues this SMF record. The name is in the form of CSFzzzz.
SMF82HPSK_SYM_LABEL_CNT	Number of SYM labels present in this record.
The following is repeated SMF82HPSK_SYM_LABEL_CNT number of times	
SMF82HPSK_SYM_LABELS	The format of the token.

SMF Record Type 82 Subtype 28

High Performance Encrypted Key

```
Subtype=001C HPSK request
Written when an attempt is made to rewrap a CCA token for encrypted-key CPACF
25 Jul 2017 21:44:22.63
   TME... 00776B57 DTE... 0117206F SID... SP21    SSI... 00000000 STY... 001C
   Name of the calling function: CSNBKRR2
   Flags = 40000000
           40000000 Rewrapping was permitted for this key
   Number of labels: 1
   Labels:
     DATASET.EYSHA.ICSF.ENCRYPT.ME.ENCRKEY.00000001                  DATA
ICSF Server Identity...
   USRI.. SYSTASK
   GRPN.. SYS1
   JBN... CSFEPC1
   RST... 13:39:45.82
   RSD... 25 Jul 2017
   SUID.. 4040404040404040
End User Identity...
```

```
End User Identity...
   USRI.. DATAOWN
   GRPN.. SYS1
   TRM... LOCALC12
   JBN... DATAOWN
   RST... 13:40:28.77
   RSD... 25 Jul 2017
   SUID.. 4040404040404040
```

Is there a way to audit crypto engine usage?

ICSF will provide crypto usage tracking of applications and components that invoke ICSF services in HCR77C1. Crypto usage tracking can be enabled/disabled at ICSF initialization using the Installation Options Data Set (IODS) or dynamically using SETICSF OPT operator commands.

ICSF IODS Option	SMF Record Type
STATS(ENG,SRV,ALG)	Type 82 Subtype 31

ENG: Tracks crypto engine usage. When enabled, ICSF tracks the usage of Crypto Express Adapters, Regional Cryptographic Servers, CPACF and Software.

SRV: Tracks crypto service usage. When enabled, ICSF tracks the usage of ICSF callable services and User Defined Extensions (UDX).

ALG: Tracks crypto algorithm usage. When enabled, ICSF tracks the usage of crypto algorithms that are referenced in cryptographic operations.

Crypto usage data collection is synchronized to the SMF recording interval. Your SMFPRMxx member must contain:

- The collection interval (INTVAL)
- The synchronization value (SYNCVAL)
- The Crypto Usage Statistics Subtype 31 for ICSF Type 82 records (TYPE)

SMF Record Type 82 Subtype 31 (Fixed Header)

Crypto Usage Statistics

Name	Description
SMF82STAT_VER	Version number
SMF82STAT_DOMAIN	Current domain index
SMF82STAT_LEN	Length of this header
SMF82STAT_TRIPL_OFF	Offset from SMF82STAT into triplet section
SMF82STAT_TRIPL_LEN	Length of triplet section
SMF82STAT_D_INTVAL_STARTE	Start time (TOD clock) of the SMF interval in STCKE format.
SMF82STAT_D_INTVAL_ENDE	End time (TOD clock) of the SMF records in STCKE format.
SMF82STAT_D_USERID_AS	The HOME address space user id
SMF82STAT_D_USERID_TK	The task level user id (if present)
SMF82STAT_D_JOBID	The job id for the HOME address space.
SMF82STAT_D_JOBNAME	The job name for the HOME address space.
SMF82STAT_D_JOBNAME2	The job name of the SECONDARY address space (ICSF caller).
SMF82STAT_D_PLEXNAME	The Sysplex member name.

SMF Record Type 82 Subtype 31 (Triplets)

Crypto Usage Statistics

Tag	Name	Description
X‘0201’	SMF82STAT_ENG_CARD	Identifier, serial number and usage count
X‘0202’	SMF82STAT_ENG_RCS	Identifier, serial number and usage count
X‘0203’	SMF82STAT_ENG_CPACF	Usage count
X‘0204’	SMF82STAT_ENG_SOFTW	Usage count
X‘0205’	SMF82STAT_SRV	Service name and usage count
X‘0206’	SMF82STAT_SRVUDX	UDX service name and usage count
X‘0207’	SMF82STAT_ALG	Algorithm name and usage count

Generally, crypto usage statistics are intended to help you determine:

- Which jobs/tasks are using the various crypto engines
- Which crypto adapter types are getting the most requests
- If any crypto requests are being handled in software
- What are the peak periods of crypto utilization
- Which ICSF services are being invoked by other z/OS components
- Which jobs / tasks are using out-of-date algorithms or key sizes

Use the STATSFILTERS(NOTKUSERID) installation options data set keyword to reduce the number of SMF records in high transaction environments.

SMF Record Type 82 Subtype 31

Crypto Usage Statistics

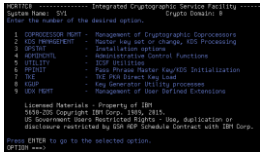
```
Subtype=001F Crypto Usage Statistics
Written periodically to record crypto usage counts
25 Jul 2017 21:44:30.00
   TME... 00776E38 DTE... 0117206F SID... SP21    SSI... 00000000 STY... 001F
   INTVAL_START.. 07/26/2017 01:43:30.005793
   INTVAL_END.... 07/26/2017 01:44:30.004008
   USERID_AS..... DATAOWN
   USERID_TK.....
   JOBID......... T0000060
   JOBNAME....... DATAOWN
   JOBNAME2......
   PLEXNAME...... LOCAL
   DOMAIN........ 0
   ENG...CARD...5C47/99EA6076... 1
   ENG...CPACF... 1
   ALG...AES256..... 1
   SRV...CSFKRR2.... 2
```

What IBM tools are available to manage keys?

Integrated Cryptographic Services Facility (ICSF)

ICSF provides callable services and utilities that generate, store, and manage keys, and also perform cryptographic operations.

Supports Master Keys and Operational Keys



Trusted Key Entry (TKE) Workstation

TKE securely manages multiple Cryptographic Coprocessors and keys on various generations of IBM Z from a single point of control.

Supports Master Keys and Operational Keys

Let's take a closer look

Enterprise Key Management Foundation (EKMF)

EKMF securely manages keys and certificates for cryptographic coprocessors, hardware security modules (HSM), cryptographic software, ATMs, and point of sale terminals.

Supports Operational Keys

Security Key Lifecycle Manager (SKLM)

SKLM v2.7 provides key storage, key serving and key lifecycle management for IBM and non-IBM storage solutions using the OASIS Key Management Interoperability Protocol (KMIP) and IBM Proprietary Protocol (IPP).

Supports Operational Keys for Self Encrypting Devices (SEDs)

z/OS Integrated Cryptographic Services Facility (ICSF)

ICSF works with the hardware cryptographic features and the Security Server (RACF element) to provide secure, high-speed cryptographic services in the z/OS environment.

- ICSF provides the application programming interfaces by which applications request cryptographic services.
- ICSF provides panels to load CCA master key values onto secure cryptographic features, allowing the hardware features to be used by applications.
- ICSF callable services and programs can be used to generate, store, and manage keys that are used in the cryptographic functions.

Cryptographic Key Data Set
• CCA Symmetric Keys
• AES, DES and HMAC

Token Data Set
• PKCS#11 Keys, Certificates
• All algorithms

Public Key Data Set
• CCA Asymmetric Keys
• RSA, ECC and Trusted Blocks

Key Management Features for z/OS ICSF

ISPF Panels

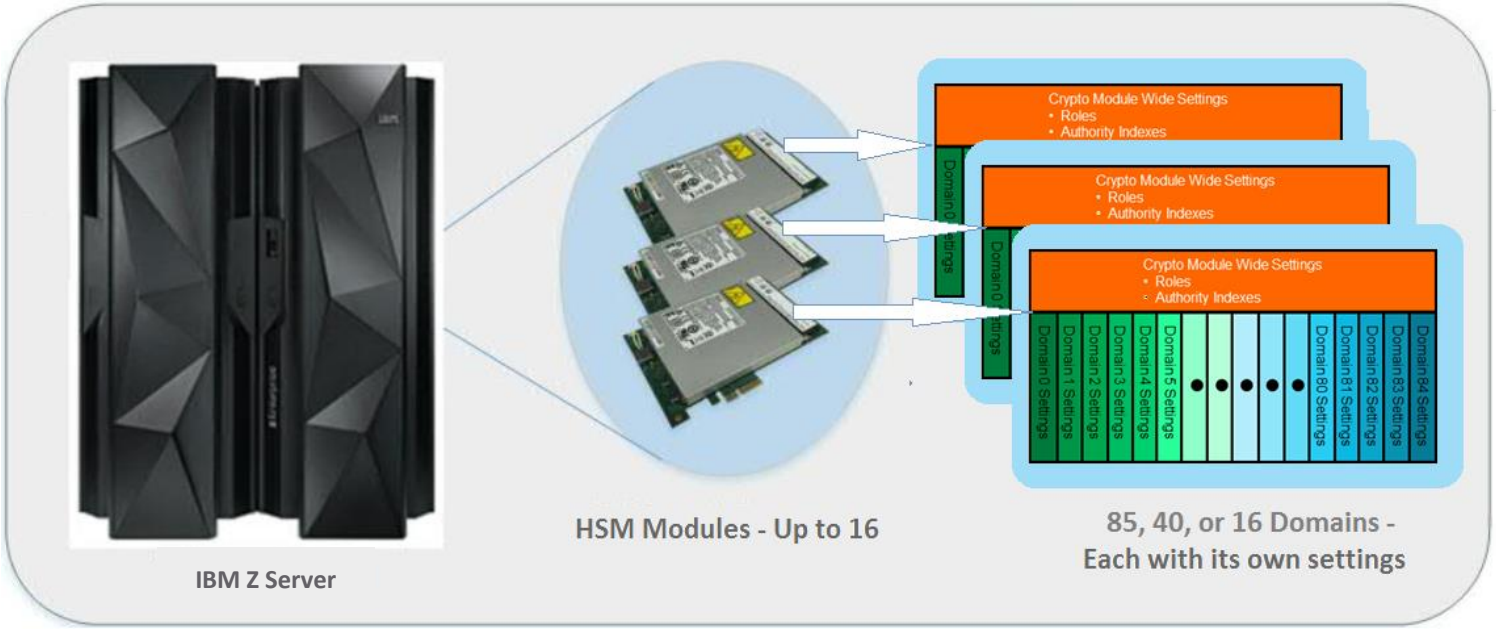
- Generate, load and view Master Keys
  - Panel 1: COPROCESSOR MGMT
  - Panel5: UTILITIES
- Manage key records in the CKDS
  - Panel 5.5: CKDS KEYS (i.e. CKDS Browser)
  - Panel 5.5.7 can generate a secure AES DATA key and store in the CKDS
- Manage key records in the PKDS
  - Panel 5.6: PKDS KEYS
- Manage PKCS #11 tokens in the TKDS
  - Panel 5.7: PKCS11 TOKEN (i.e. TKDS Browser)
- Generate keys in bulk
  - Panel 8: KGUP - Key Generator Utility Program

Application Programming Interfaces

- Create symmetric and asymmetric keys
  - CCA Symmetric Keys: CSNBKTB/2 (build key token), CSNBKGN/2 (generate key token), CSNBRNG/L (generate random numbers)
  - CCA Asymmetric Keys: CSNDPKB (build PKA key token), CSNDPKG (generate PKA key token)
  - PKCS #11 Keys: CSFPGSK (generate PKCS #11 secret key), CSFPGKP (generate PKCS #11 key pair)
- Manage Key Records in ICSF Key Data Sets
  - CKDS: CSNBKRC/2 (create), CSNBKRW/2 (write), CSNBKRR/2 (read), CSNBKRD (delete)
  - PKDS: CSNDKRC (create), CSNDKRW (write), CSNDKRR/2 (read), CSNDKRD (delete)
  - TKDS: CSFPTRC (create, copy), CSFPTRL (list), CSFPTRD (delete), CSFPGAV (get attributes), CSFPSAV (set attributes)
  - General KDS & Metadata: CSFKDSL (kds list), CSFKDMW (metadata write), CSFKDMR (metadata read)

IBM Trusted Key Entry (TKE) Workstation

TKE is an appliance that simplifies the management of IBM Z Host Cryptographic Modules running in Common Cryptographic Architecture (CCA) or IBM Enterprise PKCS#11 (EP11) mode, using compliant level management techniques.



Key Management Features for TKE

Features for Managing Module Scoped and Domain Scoped Administrative settings on Host Cryptographic Modules

- Featuring: Secure, simplified administrative management of multiple domain host cryptographic modules in complex configurations

Secure, hardware-based Master Key and Operational key management

- Featuring: Compliant level hardware-based key management with proper encryption strengths, dual controls, and security relevant auditing

Highly secure and efficient movement of administrative settings from one Host Cryptographic Module to another

- Providing: Secure, fast, and accurate deployment of new crypto modules on production, test, or disaster recovery systems

Popular Features

- Domain Grouping to broadcast a command to a set of domains
- Secure Loading of CCA Master Keys (MKs)
- Manage domains higher than 16
- Migration Wizards
- Enable/disable Access Control Points (ACPs)
- Loading MKs for inactive LPARs
- Loading PIN decimalization tables
- Loading EP11 Master Key

IBM Enterprise Key Management Foundation (EKMF)

Secure workstation

- is used for **generating** all new keys by users authenticated with **smart cards** or automatically based on requests. Workstation utilizes **IBM 4765/7**

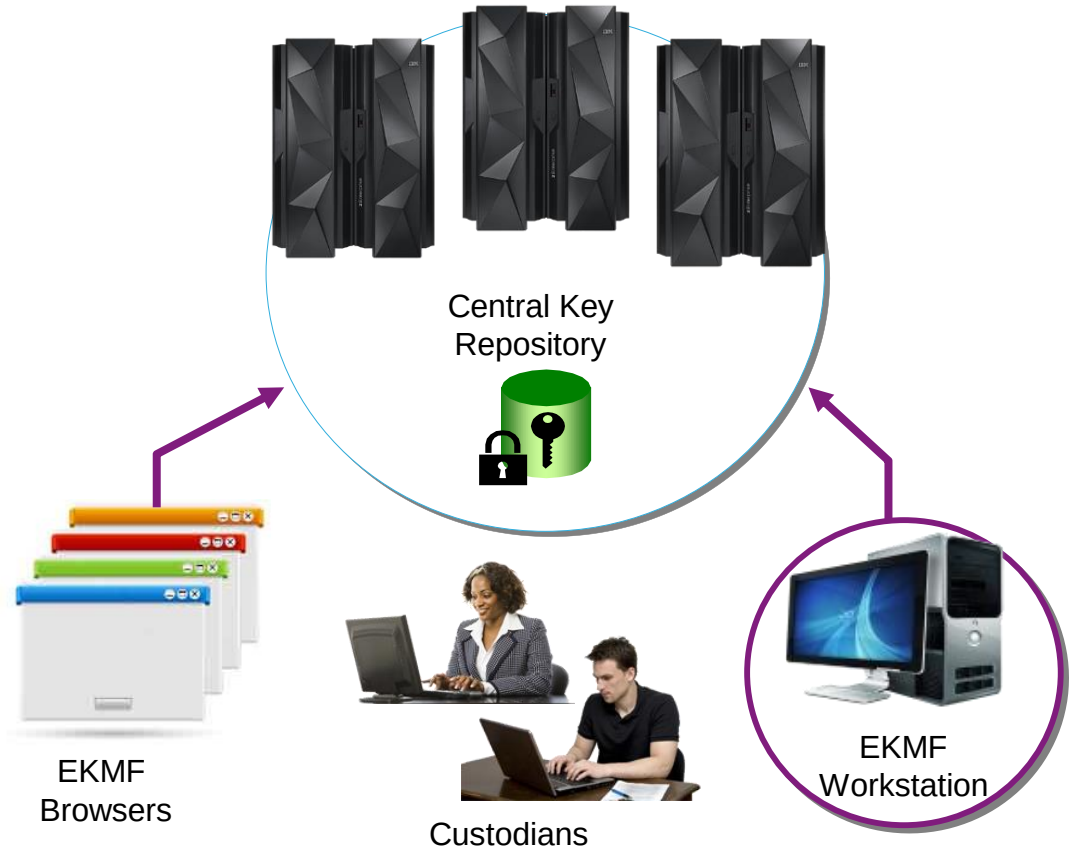
Central repository

- contains **keys** and **metadata** for all cryptographic keys produced by the EKMF workstation. This enables easy **backup** and recovery of key material.

EKMF Browser

- features **monitoring** capabilities and enables **planning** of future key handling session to be executed on the workstation.

Note that while this is a mainframe centric view, EKMF supports distributed platforms as well.



Key Management Features for EKMF

Basic key management functions include:

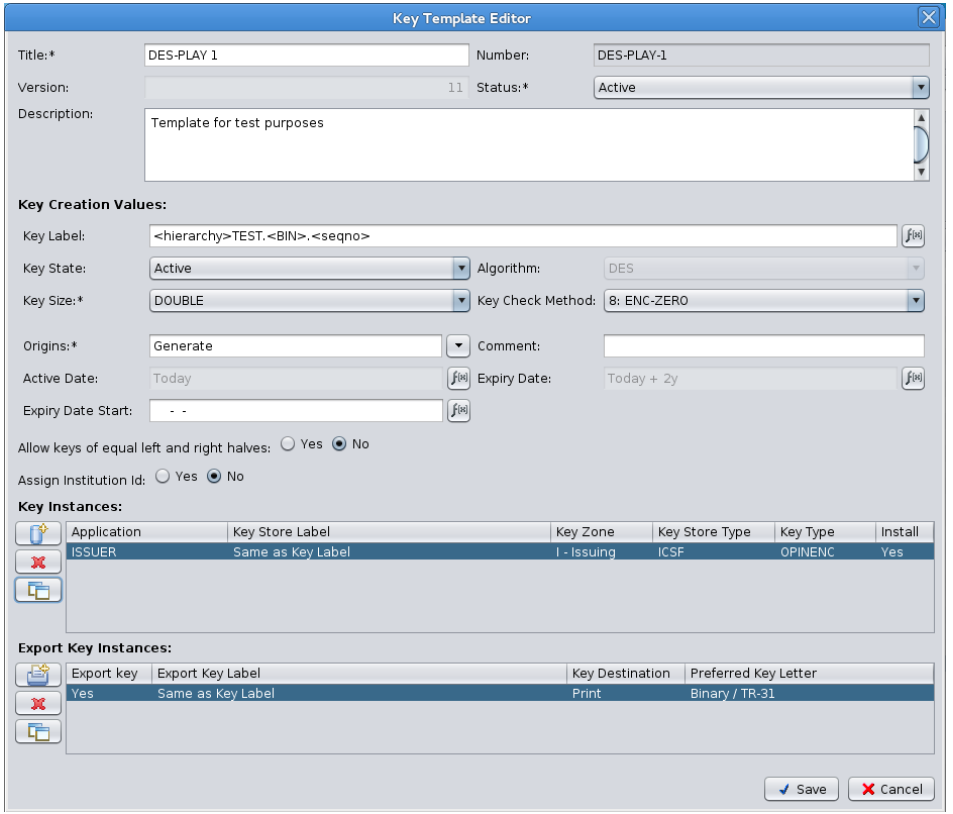
- key generation
- key import
- key export
- key print
- key administration

Key management functions are controlled by key templates and key policies. Key templates:

- control functions for a key
- predefine key attributes

When generating or entering a key, the key is automatically distributed to the servers specified in the key template.

- ICSF Key Data Sets
- RACF Key Rings (i.e. SKLM, z/OS PKI)
- … and more

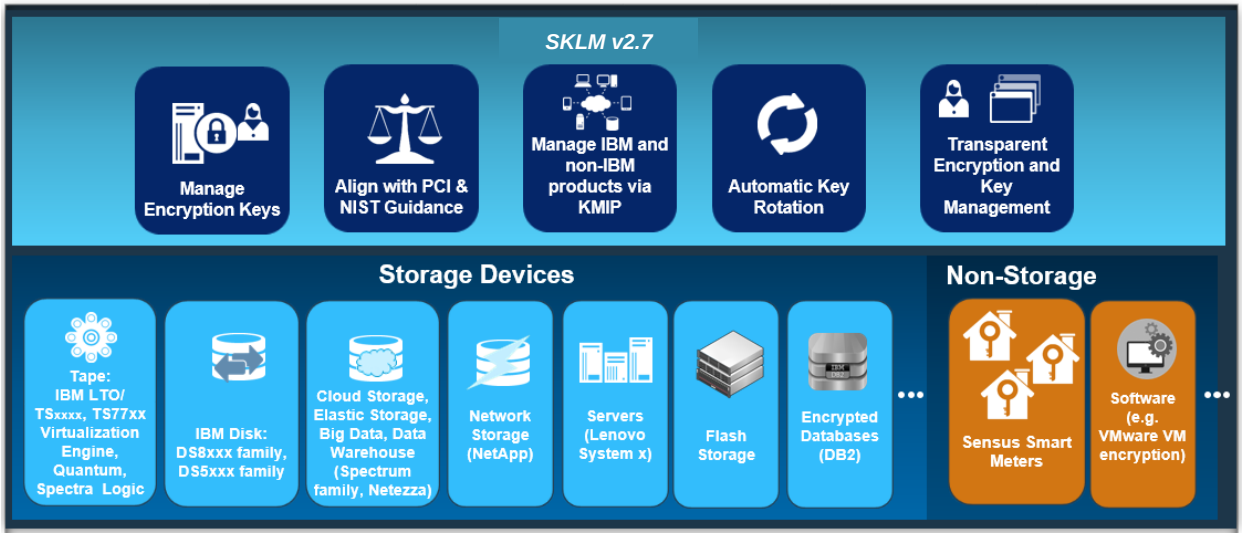


IBM Security Key Lifecycle Manager (SKLM)

IBM Security Key Lifecycle Manager provides centralized key management for self-encrypting devices.

Self-encrypting devices protect data if you lose control of the device.

- Data on the truck traveling between datacenters
- Data at rest within the datacenter
- Decommissioned storage devices



Key Management Features for SKLM

SKLM for Distributed Systems

SKLM v2.7 supports the IBM Proprietary Protocol (IPP) and industry-standard Key Management Interoperability Protocol (KMIP) for key distribution with storage devices.

Features include:

- Key generation, import and export
- Secure storage of key material
- Automatic assignment and rotation of keys
- Key serving at the time of use

SKLM for z/OS

SKLM for z/OS supports the IBM Proprietary Protocol (IPP) for key distribution with storage devices.

SKLM for z/OS can use ICSF through JCE hwkeytool or RACF GENCERT commands to push RSA key pairs to the ICSF PKDS and AES keys to the ICSF CKDS.

Features include:

- Key generation, import and export
- Secure storage of key material
- Key serving at the time of use

Note: SKLM can not be used to manage z/OS data set encryption keys.

Key Management Activities

SEDs = Self-encrypting devices

	Activity	ICSF	TKE	EKMF	SKLM
Authorization Tasks	SAF Authorization (CSFKEYS and CSFSERV)	YES	YES	YES	SKLM for z/OS
	Key Auditing (master keys, operational keys)	YES	YES	OPERATIONAL KEYS	YES
Master Key Tasks	Master Key Entry	YES, PANELS	YES, SECURE	NO	NO
	Master Key Change	YES, PANELS	YES, SECURE	NO	NO
	Master Key Zeroize	NO, HMC / SE	YES	NO	NO
Basic KDS Tasks	Operational Key Record Creation (and naming)	YES	NO	YES, GUI-BASED	SEDs
	Operational Key Record Update	YES	NO	YES, GUI-BASED	SEDs
	Operational Key Record Deletion	YES	NO	YES, GUI-BASED	SEDs
Basic Key Tasks	Operational Key Generation	YES	SMALL SCALE	YES, GUI-BASED	SEDs
	Operational Key Import	YES	SMALL SCALE	YES, GUI-BASED	SEDs
	Operational Key Export	YES	NO	YES, GUI-BASED	SEDs
KDS Metadata Tasks	Operational Key Archival	YES	NO	NON-KDS,GUI-BASED	NO
	Operational Key Restore	YES	NO	NON-KDS,GUI-BASED	NO
	Operational Key Expiration	YES	NO	NON-KDS,GUI-BASED	NO
Maintenance Tasks	Rekeying encrypted data (operational keys)	YES	NO	NO	SEDs
Recovery Tasks	Disaster Recovery (master keys, operational keys)	YES	YES	OPERATIONAL KEYS	SEDs

Appendix: Key Rotation

How do you rotate keys?

There are two types of key rotation that you can perform on IBM Z:

- Master Key Rotation
- Operational Key Rotation

Master Keys

Master keys are used only to encipher and decipher keys.

Master keys are stored in secure, tamper responding hardware.

Operational Keys

Operational keys are used in various cryptographic operations (e.g. encryption).

Operational keys may be stored in a key store (e.g. data set, file, database) or returned back to the caller.

Operational keys may be encrypted by a Master Key to be considered secure keys.

How does Master Key Rotation work?

Master key rotation involves re-enciphering secure, operational keys that reside in Key Data Sets. Re-encipherment occurs in the secure boundary of the Crypto Express adapter. ICSF synchronizes the changes across members of the sysplex sharing the same Key Data Set (when applicable).

For each secure key:
- The operational key value is decrypted from under the current Master Key
- The operational key value is encrypted with the new Master Key

After all secure keys have been re-enciphered:
- The current Master Key becomes the old Master Key
- The new Master Key becomes the current Master Key

Using Coordinated Change MK, the master key rotation is non-disruptive. Master keys can be rotated while crypto workloads are running.

Master Key Rotation Procedure

1. Allocate new Key Data Sets.
2. Generate and load new Master Keys using TKE or ICSF. (You must load the same MK on all sysplex members sharing the KDS.)
3. Initiate the Coordinated Change MK (CCMK) operation using TKE or ICSF

Note: CCMK can be run on a single system as well as a sysplex.

Example data set allocation for the CKDS…

The **current / active key data set** containing the existing keys could be EYSHA.ICSF.CSF77C1.20180101.CKDSR

The **new key data set** to contain the re-enciphered keys could be EYSHA.ICSF.CSF77C1.20190101.CKDSR

Trusted Key Entry (TKE) Workstation

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

             AES new master key register       :  EMPTY
             DES new master key register       :  EMPTY
             ECC new master key register       :  EMPTY
             RSA new master key register       :  EMPTY

Specify information below

  Key Type  ===> AES-MK           (AES-MK, DES-MK, ECC-MK, RSA-MK)

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

  Checksum  ===> 42

  Key Value ===> 24BF3F412727DA29
            ===> 17DF1B161A04E7B9
            ===> 10AD680264CA686A   (AES-MK, ECC-MK, and RSA-MK only)
            ===> 583835BFA1288930   (AES-MK, ECC-MK only)

Press ENTER to process.
```

ICSF Master Key Entry Panel

Let's take a closer look

Master Key Rotation affects Crypto Express Adapters and Key Tokens

Crypto Express6S

1. Load

32-byte AES Master Key

```
2F44E4823BCD2852
8D0F3F9EB109A9A9
E57C507CF7D684C5
ED9A9AD68D0E9B12
```

AES Master Key Verification Pattern (MKVP)

```
F506FF13A5FFC26A
```

2. Initiate Coordinated Change MK

64-byte AES DATA Secure Key Token

```
010000000400C089F506FF13A5FFC26A
1059E00CA93B5876F0A6B6E45E3835E5
A4AC94D47D52A8781BDDF9BF4CA32F51
00000000000000000100002033FC0DCD
```

3. Re-encipher key
4. Update MKVP
5. Store in new data set

CKDS

```
F506FF13A5FFC26A
```

AES MKVP

Ciphertext is changed

DATASET.MEDICAL.M1.0001

DATASET.MEDICAL.M1.0002

32-byte AES DATA Clear Key Value

```
ABC82AE086AE237FEAC3DFD49091E15E
E26B98B76784F4227CC32C0B907184A6
```

Underlying Clear Key Value is unaffected

How do you rotate operational keys? Two approaches!

Approaches	Policy-Based Data Set Encryption Example	Considerations

"Aging Out"	• Data sets were encrypted using Operational Key DATASET.MEDICAL.G1.0001
	• At some interval, the ICSF admin generates DATASET.MEDICAL.G1.0002
	• The security admin updates associated DATASET resources to use key label DATASET.MEDICAL.G1.0002 from that point forward.
	  • DATASET.MEDICAL.G1.0001 is still in use by old / existing data
	  • DATASET.MEDICAL.G1.0002 is used for new data
		• Not sufficient when a key has been compromised
		• Affects new data only
		• Existing data is not re-encrypted
		• Old keys must remain in the CKDS
		• More keys to manage
		  • Key sequence numbers are recommended

Re-encrypt all data	• Data sets were encrypted using Operational Key DATASET.MEDICAL.G1.0001
	• At some interval, the ICSF admin generates DATASET.MEDICAL.G1.0002
	• The security admin updates associated DATASET resources to use key label DATASET.MEDICAL.G1.0002 from that point forward.
	  • DATASET.MEDICAL.G1.0002 is used for new data
	• All data encrypted with DATASET.MEDICAL.G1.0001 must be identified and re-encrypted
	  • DATASET.MEDICAL.G1.0001 would no longer be in use
		• Recommended when a key has been compromised
		• Affects all data (new and existing)
		• Must identify ALL data encrypted with the old key
		• Archiving the old key is recommended over deleting the old key
		• Crypto-periods may be established to restrict key usage.

Operational Key Rotation Procedure – “Aging Out”

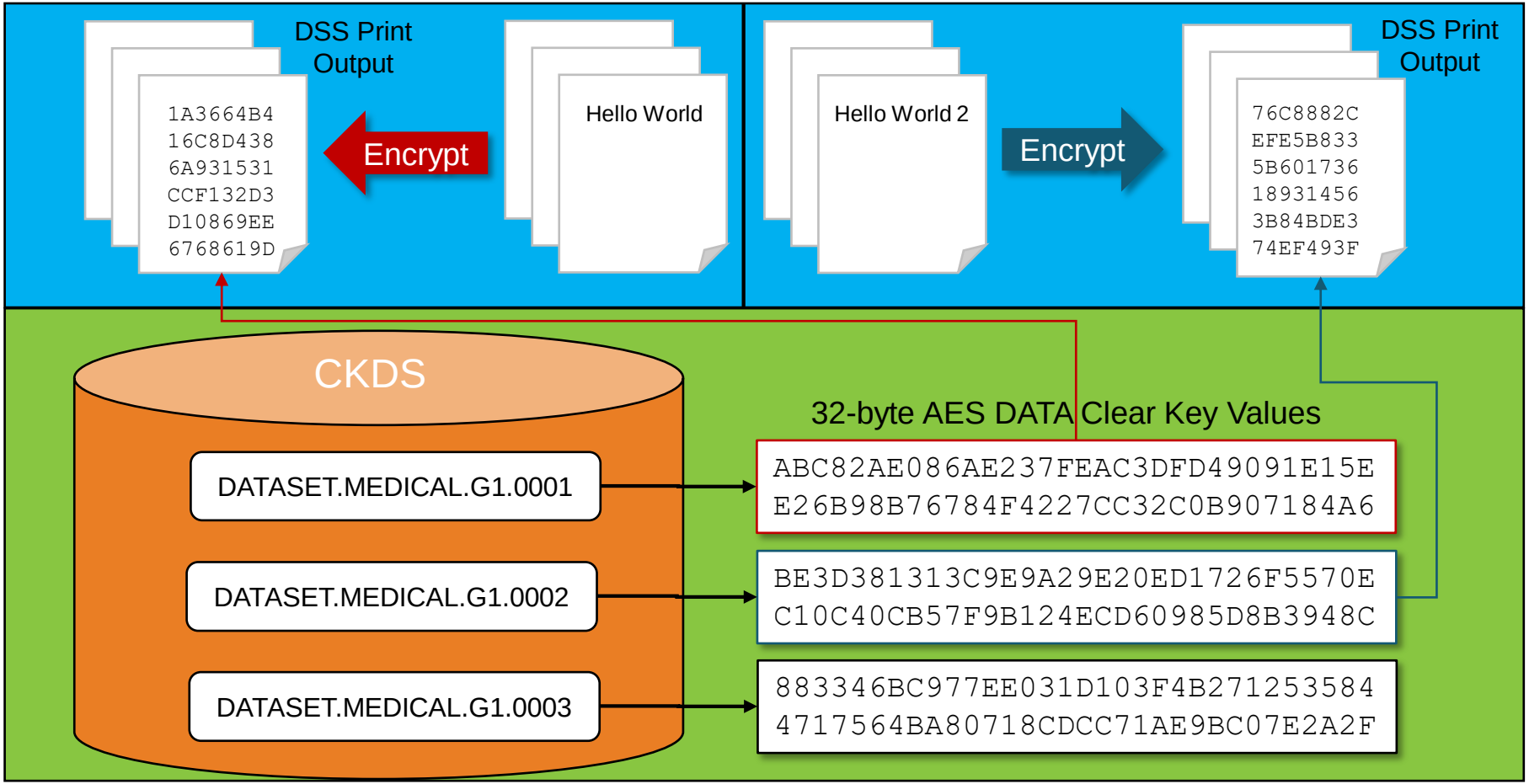
For this example procedure, DATASET.MEDICAL.G1.0001 had been used to encrypt some data sets.

1. Generate a new operational key, DATASET.MEDICAL.G1.0002
2. Locate all DATASET profiles associated with DATASET.MEDICAL.G1.0001
3. Update the DATASET profiles with the new key label, DATASET.MEDICAL.G1.0002

All newly allocated datasets will use DATASET.MEDICAL.G1.0002.

Operational key rotation by “Aging out” is non-disruptive. Existing workloads can continue to run while the key is rotated.

Operational Key Rotation affects Encrypted Data - “Aging Out”



Operational Key Rotation Procedure – Re-encrypt All Data

For this example procedure, DATASET.MEDICAL.G1.0001 had been used to encrypt some data sets.

Part 1 (Same process as "Aging Out")

1. Generate a new operational key, DATASET.MEDICAL.G1.0002
2. Locate all DATASET profiles associated with DATASET.MEDICAL.G1.0001
3. Update the DATASET profiles with the new key label, DATASET.MEDICAL.G1.0002

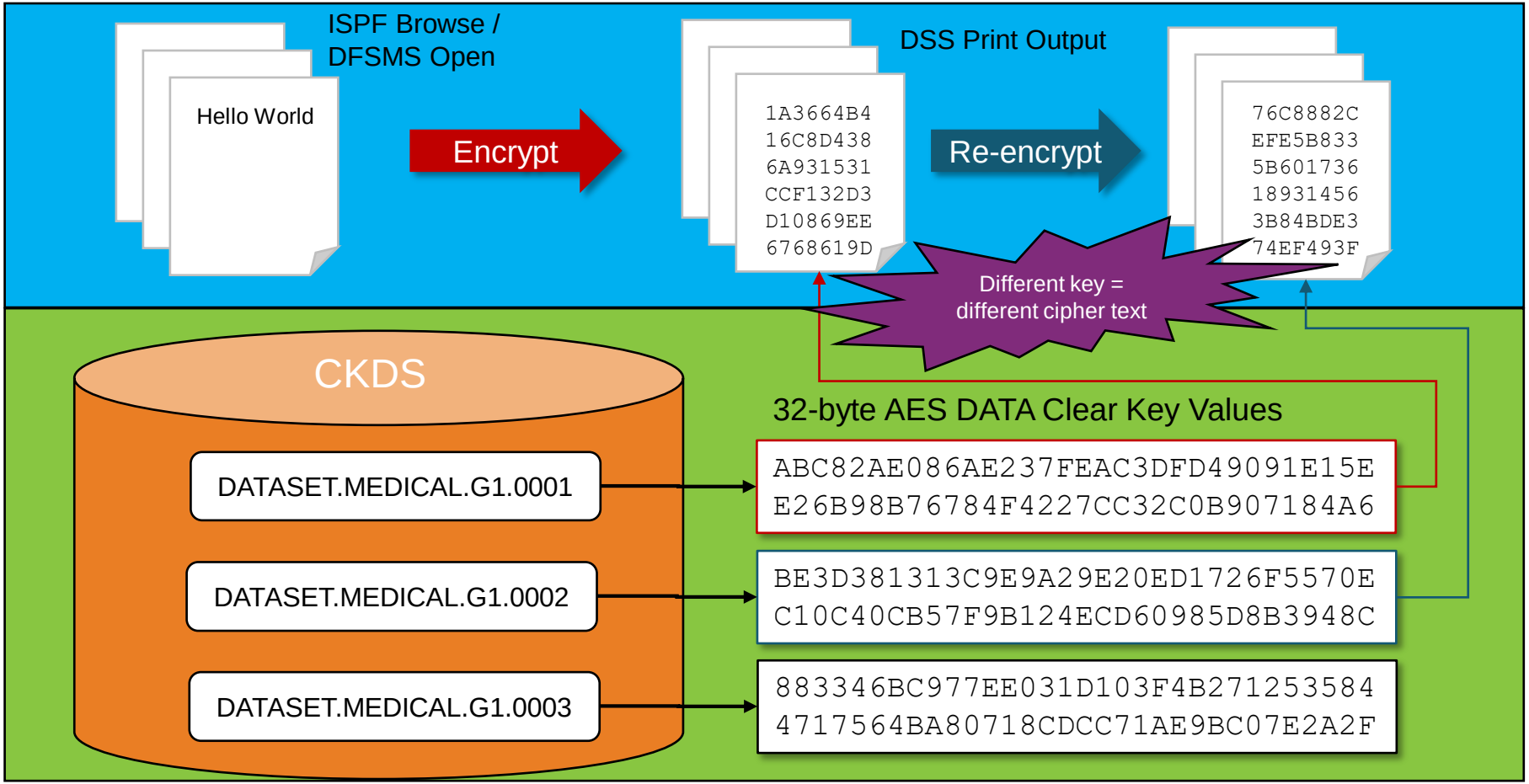
Part 2 (Re-encrypt existing data sets)

1. Identify all data sets encrypted with old key label DATASET.MEDICAL.G1.0001
   - DASD and Tape
   - Migrated and Active
   - …
2. Allocate new data sets covered by DATASET profiles associated with DATASET.MEDICAL.G1.0002
3. Copy the data from the old data sets to the new data sets
   - The user performing this operation requires access to DATASET.MEDICAL.G1.0001 and DATASET.MEDICAL.G1.0002 for the duration of the operation.
4. Delete the old data sets
5. Archive the old key, DATASET.MEDICAL.G1.0001

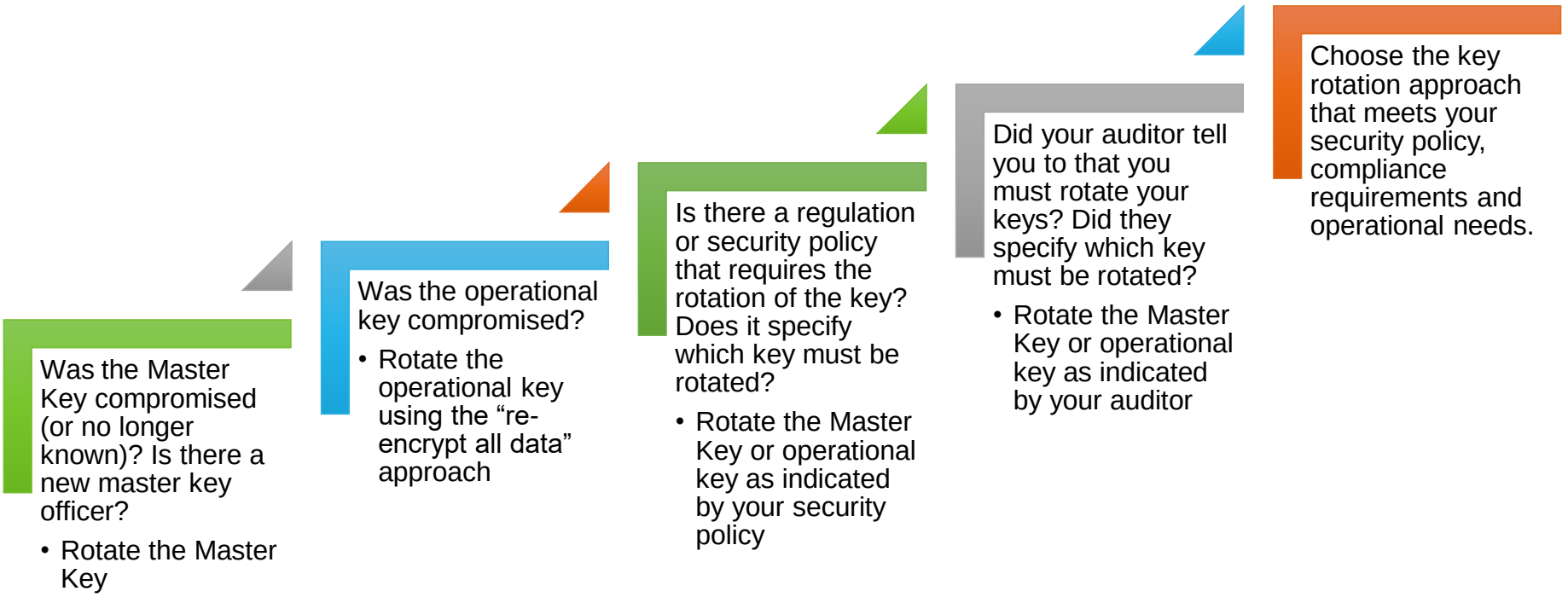
All data sets that had been encrypted with DATASET.MEDICAL.G1.0001 are now encrypted with DATASET.MEDICAL.G1.0002.

Operational key rotation by re-encrypting all data is typically disruptive. If you are running Db2 workloads, you can initiate an online reorg to make the re-encryption process non-disruptive. For other workloads, you will need to stop the workload while the key is being rotated.

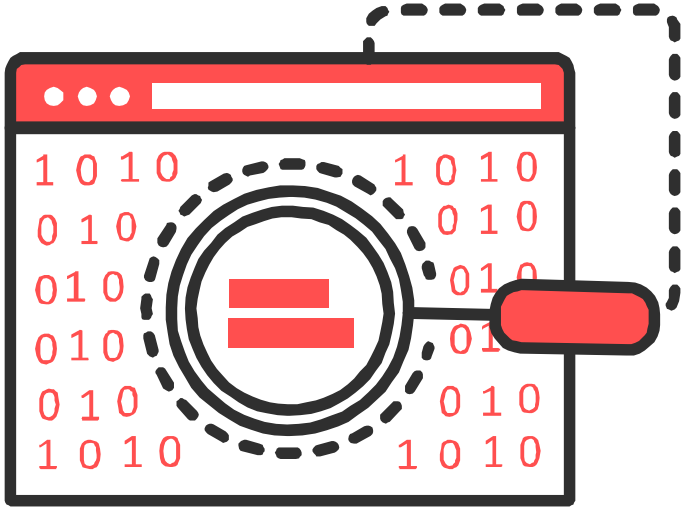
Operational Key Rotation affects Encrypted Data – Re-encrypt All Data



How do you decide which key rotation approach to use?



Questions?



Additional Resources

IBM Crypto Education Community
https://www.ibm.com/developerworks/community/groups/community/crypto

Master Key Management Materials
https://ibm.biz/BdiKRz