IBM Advanced Technical Support - Americas

z/OS Key Management for Tape Encryption

CRP 7 Part 1: EKM Providers RACF and ICSF with Crypto Hardware.
CRP 8 Part 2: EKM JCE Provider with Crypto Hardware

Marilyn Frazier Allmond
Vanguard 2008
Washington Systems Center

ON DEMAND BUSINESS™

Disclaimers and Trademarks 1 of 2

- **Copyright© 2006 by International Business Machines Corporation.**
- **No part of this document may be reproduced or transmitted in any form without written permission from IBM Corporation.**
- **The performance data contained herein were obtained in a controlled, isolated environment. Results obtained in other operating environments may vary significantly. While IBM has reviewed each item for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. These values do not constitute a guarantee of performance. The use of this information or the implementation of any of the techniques discussed herein is a customer responsibility and depends on the customer's ability to evaluate and integrate them into their operating environment. Customers attempting to adapt these techniques to their own environments do so at their own risk.**
- **Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This information could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or programs(s) at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only**
- **References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectually property rights, may be used instead. It is the user's responsibility to evaluate and verify the operation of any on-IBM product, program or service.**

Disclaimers and Trademarks 2 of 2

- **THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT.**
- **IBM shall have no responsibility to update this information. IBM products are warranted according to the terms and conditions of the agreements (e.g. IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. IBM is not responsible for the performance or interoperability of any non-IBM products discussed herein.**
- **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 in connection with this publication 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.**
- **The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:**

  IBM Director of Licensing
  IBM Corporation
  North Castle Drive
  Armonk, NY 10504-1785
  U.S.A.
- **The following terms are trademarks or registered trademarks of the IBM Corporation in either the United States, other countries or both.**
  - IBM, TotalStorage, zSeries, pSeries, xSeries, S/390, ES/9000, AS/400, RS/6000
  - z/OS, z/VM, VM/ESA, OS/390, AIX, DFSMS/MVS, OS/2, OS/400, ESCON, Tivoli
  - iSeries, ES/3090, VSE/ESA, TPF, DFSMSdfp, DFSMSdss, DFSMShsm, DFSMSrmm, FICON,
- **Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both. Other company, product, and service names mentioned may be trademarks or registered trademarks of their respective companies.**
- **VeriSign and the VeriSign Secured Design is a registered trademark of the VeriSign Inc. The Checkmark Circle Design is a trademark of VeriSign Inc.**

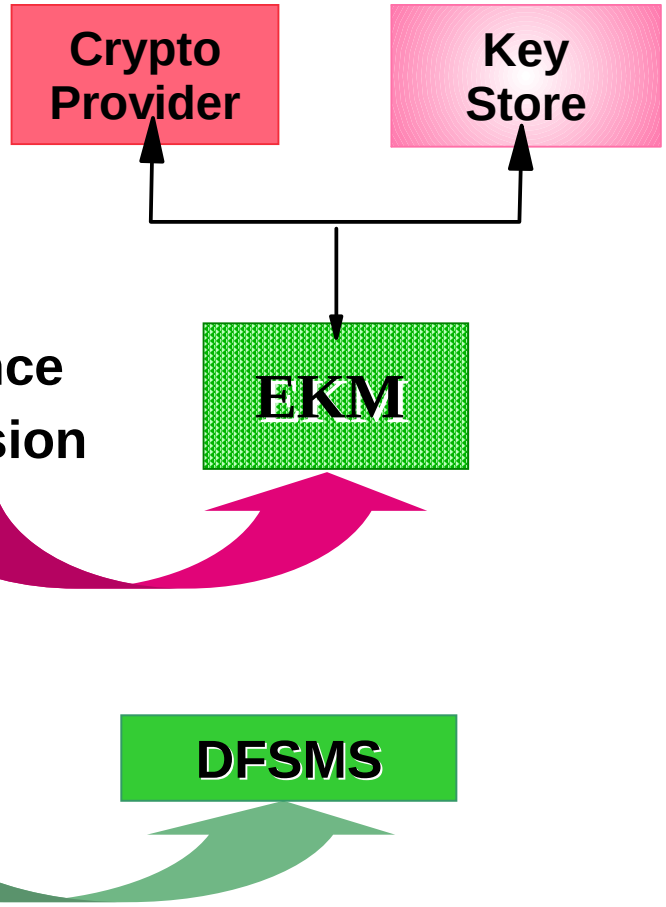
Focus

- **Solely on the aspect of the Encryption Key Management on z/OS for Tape Encryption. This presentation will not discuss setup details for Tape Encryption but will focus on the generic issues that should be considered when choosing where to implement an EKM.**

- **For this presentation it is more important to go deeper into the key management issues, options, and the impact these encryption choices might mean in a productive environment.**
  - Crypto provider(s) and key stores
  - DR impact
  - Archival Requirements

Encrypting Tape System for System z

- **CRP 7 Part 1 and 2 deal with 1 of the major elements and briefly speak to the other**
  - **Encryption Key Management**
    - Defines how the cryptographic key management is done
      - **Generation**
      - **Control**
      - **Maintenance**
      - **Transmission**
  - **Encryption Policy**
    - Defines rules that govern which volumes are encrypted
    - Defines the mechanism for key selection



Encryption Policy via DFSMS

- **Data class policy specifying EEFMT2**
  - If encryption-capable TS1120 tape drive is allocated, the default recording format is EFMT2
- **Modify or create ACS routines**
- **DD statement keyword parameters** (data class parameters)
  - KEYLABL1 and KEYLABL2
    - Specifies the label for the key encrypting key (KEK) used by the EKM to encrypt the data (encryption) key
    - 1- 64 characters
  - KEYENCD1 and KEYENCD2
    - Specifies how the label for the KEK specified by the key label is encoded by the EKM and stored on the tape cartridge
    - 1 character
      - L = label
      - H = public key hash

z/OS Tape Encryption Key Management

- **Key Management consists of a number of elements including:**
  - A key store where the keys to encrypt and decrypt the data key are securely kept, such that they are highly available, and a place to where certificates are held.
  - An Encryption Key Manager (EKM) that communicates with the drive through the designated key management path (in-band or out-of-band).
  - Ability of the tape drive to request and process (with the Encryption Key Manager) key management information.
  - A management interface for configuring and managing the Encryption Key Manager (EKM).

z/OS Tape Encryption Key Management . . .

- **In-band proxy**
  - allows key management information to be exchanged with a tape drive over existing ESCON/FICON
  - z/OS proxy interface communicates with the tape drive (through the control unit) in the current data path and then uses a TCP/IP connection to communicate with the Encryption Key Manager.
- **Management interfaces**
  - configure whether to use a direct TCP/IP connection between the storage devices and the Encryption Key Manager (out-of-band) or
  - configure to use the in-band proxy.
  - command line interface client.

Encryption Key Manager (EKM)

- **Encryption Key Manager (EKM) is the application that is part of the IBM Java environment.**
- **EKM acts as a background process waiting for key generation or key retrieval requests sent to it via a TCP/IP path between it and the tape library, controller, subsystem, device driver, or tape drive.**
- **For TS1120 Tape Drives EKM generates an Advanced Encryption Standard (AES) key and gives it to the tape drives in two forms:**
  - Encrypted or *wrapped*, RSA key pairs
  - Separately wrapped for secure transfer to the tape drive
- **For LTO Ultrium 4 Tape Drives EKM fetches an existing AES key from a keystore & wraps it for secure transfer to the tape drive.**

Encryption Tape Manager on System z

- **EKM has three main components used to control its behavior.**
  - **Key Store**
    - A keystore holds the certificates and keys (or pointers to them) used by EKM to perform cryptographic operations.
    - Several types of keystores are supported and are dependent on the crypto provider(s) selected
  - **Configuration Files**
    - Customize the behavior of EKM to meet site needs
  - **Tape Drive Table**
    - Lists the tape devices EKM supports.
    - The tape drive table is a non-editable, binary file whose location is specified in the configuration file.

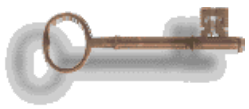
EKM Main Components

- **Java security keystore**
  - Keystore is defined as part of the Java Cryptography Extension (JCE) and an element of the Java Security components, which is part of the Java runtime environment.
  - A keystore holds the certificates and keys used by EKM to perform cryptographic operations.
  - Four (4) keystores available on System z
- **Configuration file**
  - EKM configuration file allows you to tailor the behavior of EKM to meet the needs of your organization.
  - 2 files : 1 for the server, KeyManagerConfig.properties, and 1 for the client
  - These behavioral choices are described in EKM Introduction, Planning, and User's Guide, GA76-0418 Appendix B

EKM Main Components . . .

- **Tape drive table**
  - The tape drive table is used by EKM to keep track of the tape devices it supports.
  - The tape drive table is a non-editable, binary file whose location is specified in the configuration file.

Keystore for certificates and keys



Configuration File

KeyManagerConfig.properties

EKM PARMS

Tape Drive Table

drivesn1

drivesn2

. . .

drivesnn

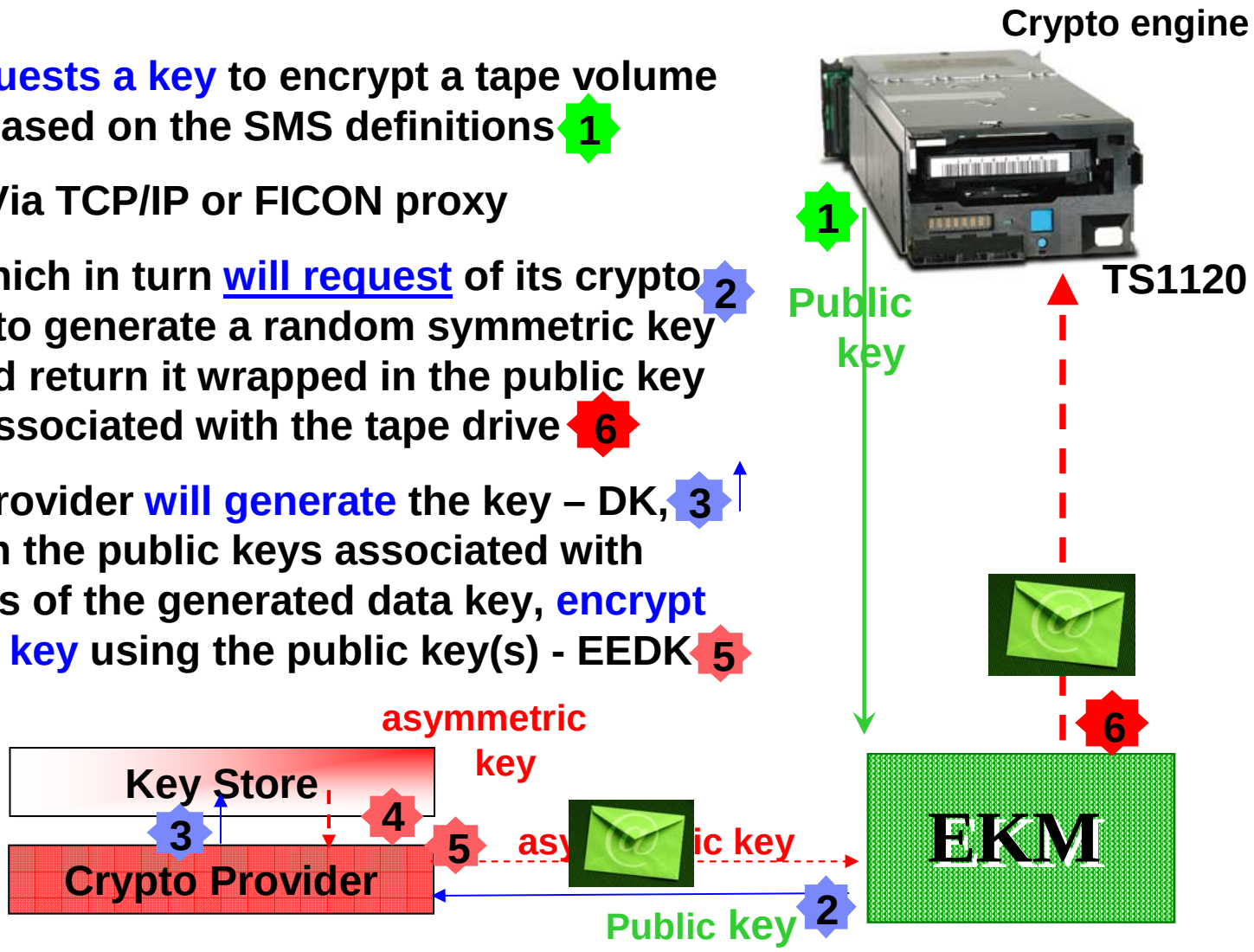
Tape Encryption Request Overview

TS1120 requests a key to encrypt a tape volume based on the SMS definitions 1

Via TCP/IP or FICON proxy

To EKM which in turn will request of its crypto 2 **provider to generate a random symmetric key value and return it wrapped in the public key associated with the tape drive** 6

Crypto Provider will generate the key – DK, 3 4 **obtain the public keys associated with recipients of the generated data key, encrypt the data key using the public key(s) - EEDK** 5



Tape Encryption Request Overview . . .

- **The tape drive will have a pair of asymmetric keys, public and private, and a certificate (generated key pair for each mount)**
- **The public drive key will be sent to the EKM in a certificate so the EKM can validate that it is communicating with the drive, i.e., authentication occurs for exchanges with the drive.**
- **The symmetric key generated by the crypto provider is "wrapped" or protected by the public key of the drive. Thus, only the drive can access the key value that will be used to encrypt the data on the tape.**
- **The symmetric key may also be encrypted by another asymmetric key for another party to receive a copy of the tape data. That asymmetric key would have been provided by the party and stored in the keystore.**
- **Asymmetric keys used to protect the symmetric key are Key-Encrypting-Keys (KEKs); the symmetric key is the Data Key (DK); and the wrapped key is the EEDK, the encrypted KD is the Session Encrypted Data Key (SEDK)**

zSeries & z/OS IBM JCE Providers Overview

- **There are 2 z/OS Java providers that can be used by EKM**
  - IBMJCE
  - IBMJCECCA
- **To use a keystore type that contains “4758” or “CCA” you must have the crypto hardware available**
- **See the URL below for a discussion and overview of zSeries hardware cryptography and JCE.**

  http://www.ibm.com/servers/eserver/zseries/software/java/jcecca14.html
- **Each provider has a keystore or a selection of keystores**
- **Keystore selection is based on environment, usage and disaster recovery requirements**

zSeries & z/OS KeyStore Overview

- **Which keystore?**
  - Repository choice for keys and certificates depends on site operational needs
  - Can be separate from hardware CPACF, CEX2C & PKDS
- **z/OS platform supports these 4 keystores**
  - JCEKS (UNIX file based)
  - JCE4758KS/JCECCAKS (ICSF, Secure hardware, & keystore for certificates)
  - JCE4785RACFKS/JCECCARACFKS (RACF with secure crypto hardware)
  - JCERACFKS (RACF/SAF)

Note: KS = key store, CCA = Common Crypto Architecture 4758 = a CCA hardware device RACF = Resource Access Control Facility SAF = System Authorization Facility

zSeries & z/OS KeyStore Overview . . .

- **Of those four options only 1 is a file-based keystore, JCEKS, and is used with IBMJCE provider discussed in Part 2**
- **The other 3 options are provided by the IBMJCECCA**
- **IBMJCECCA provider supports these keystores:**
  - JCECCAKS replacing JCE4758KS,
  - JCECCARACFKS replacing JCE4758RACFKS
  - JCERACFKS
- **IBMJCECCA is integrated into the Java 2 SDK Version 1.4 and later. IBMJCE is the IBM provider for the JCA and JCE framework. It is fully compatible with Sun's JCE 1.2.2.**

zSeries & z/OS IBMJCECCA Provider KeyStores Overview

1) **JCECCAKS and hwkeytool utility for symmetric key**
   - uses strong encryption (triple DES) to protect the keys when stored in a file-based keystore.
   - extends and replaces the JCE4758KS
   - supports migration of keys from other keystores using normal keystore interfaces

Keystore for certificates

ICSF & CEX2C For Private Keys

Key Mgmt

hwkeytool utility & ICSF

zSeries & z/OS IBMJCECCA Provider KeyStores Overview . . .

RACF for certificates

ICSF & CEX2C For Private Keys

Key Mgmt

RACF & ICSF

2) JCE4758KS

- provides for downward compatibility
- keystore handles keys (certificates) stored in RACF keyrings.
- Replaced by the JCECCAKS

3) JCE4758RACFKS

- provides for downward compatibility

zSeries & z/OS IBMJCECCA Provider KeyStores Overview . .

JCE4758KS/JCECCAKS (Certificates in a file (keyring) and keys protected by ICSF) is a file-based keystore supported on the z/OS platform only.

- Created and managed through the JVM **hwkeytool** command and is capable of creating ICSF key entries.
- Hwkeytool ***hardwaretype*** option specifies the type of key pair being generated (CLEAR, PKDS, RETAINED).
  - CLEAR = key placed in Java keystore file, no ICSF
  - PKDS = key placed in ICSF PKDS protected by ICSF & hw
  - RETAINED = key placed in Crypto HW engine can't be removed
- Hwkeytool is the utility to manage the file-based keystore. It has parameter options to support private key storage with the zSeries Crypto hardware and ICSF.

See http://www-03.ibm.com/servers/eserver/zseries/software/java/API_users_guide.html#AppF

zSeries & z/OS IBMJCECCA Provider KeyStores Overview . . .

JCE4758RACFKS/JCECCARACFKS (Certificates stored in RACF & keys protected by ICSF)

- SAF keyring-based keystore supported on z/OS only.
  - SAF is the acronym for System Authorization Facility and is part of the operating system
- Provides SAF digital certificate support, enabling the IBMJCE and IBMJCECCA providers to retrieve keys and certificates from SAF. Managed through the **RACDCERT** or equivalent SAF command or service
- This keystore uses certificates are stored in SAF keyrings but actual private key material (value) is stored in ICSF PKDS
  - When stored in the PKDS the private key value **can** be protected by the Host PKA or Asymmetric Master Key residing in the secure crypto hardware.
  - Command options are ICSF and PCICC

zSeries & z/OS IBMJCECCA Provider KeyStores Overview . . .

IMPORTANT FACT for IBMJCECCA provider: ICSF knows nothing about

- Certificates,
- Keyrings,
- Traditional keystores

ICSF knows about its own callable service interfaces, its own key data sets (CKDS and PKDS), and its interface with the Crypto hardware (CPACF, CCF, PCIXCC, CEX2C)

Storage of PKA Private key within secure hardware

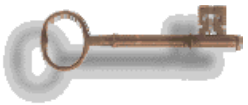
- Prevents extraction of key value for use later
- Can restrict execution options

zSeries & z/OS IBMJCECCA Provider KeyStores Overview . . .

JCERACFKS (Certificates and key material stored in RACF)

- SAF keyring-based keystore supported on z/OS only.
- Keystore can be created and managed through the **RACDCERT** or equivalent SAF command
- This keystore uses certificates generated in RACF/SAF and the key material is stored in RACF/SAF.

RACF for certificates & keys



Key Mgmt

RACF

What You Need to Know

- **If CCA is used as a keystore, keep these facts in mind:**
  - The private key will not be available once it is stored into the secure crypto hardware or the ICSF PKDS.
  - The private key is removed from all keyrings.
  - If private key needs to be kept, must store in separate media before key is moved into crypto protection.
- **Why would private key be needed**
  - To use with Disaster Recovery site EKM if separate
  - To use with secondary EKM
- **Should not place private key into crypto hardware**

Just How Much Does ICSF Do?

- **Can keep the Private and Public RSA keys in the PKDS (Public Key Data Set), when RACF or the hwkeytool might use ICSF**
  - To get the RSA key(s) into the PKDS the following ICSF API was **probably** used
    - Create a RSA key record in the PKDS, using the key label to be specified in the EKM configuration file **(CSNDKRC)**
    - Write the updated PKA record to the PKDS **(CSNDKRW)**
- **ICSF might be used to create a key value either by**
  - Key generate function **(CSNBKGN)** or
  - Random number generate **(CSNBRNG)**
- **To create the hash for key recognition, CSNBOWH**

Just How Much Does ICSF Do? . . .

- **The public key is only used to protect other keys as they are sent to the intended recipient who has the matching private key**
  - By using the public key no one other than the intended receiver can decrypt the key value being sent unless the private or public key had been compromised
  - The public key is verified for use by checking the signature(s) of the certification authority, **CSNDDSV** is used
  - For generating a signature on a self-signed certificate, **CSNDDSG** would be used
  - If the key is supplied, **CSNDPKD** could be used to decrypt it from protection by a public key and **CSNDPKE** could be used to encrypt a clear key value under an RSA public key.
    - CSNDPKD can use either a clear RSA key or a protected one

Just How Much Does ICSF Do? . . .

- **The public key is only (continued . . .)**
  - For creation of a symmetric key value, ICSF could be requested to generate a random number **(CSNBRNG)**
  - The random number is changed into a key value that can be produced by both receiver and sender and it can be either
    - Imported into a key value protected by the public key of the intended receiver **(CSNDSYI)**, OR
    - Exported into a key value protected by the public key of the intended receiver **(CSNDSYX)**
- Most likely the public key is used with CSNDSYG
  - 24-byte key is generated
  - EKM transforms to 32-bytes of AES key material

Just How Much Does ICSF Do? . . .

- **The private key is used for (continued . . .)**
  - Decrypting data key and session key from encipherment under the public key associated with the tape drive and “unwrapping”the key value
  - Wrapping Data Key and/or Session Key with encryption under public key and “unwrapping” it with the private key
  - None of the symmetric key values are stored within the tape header in the clear. They will be encrypted. The values encrypting those keys must remain with the data so that no data could be unrestorable as long as the public and/or private key values have not changed or been removed from key storage .
- **Data Key use is with AES using temporary values not stored in the keystore nor in ICSF’s CKDS**

Perils and Hidden Traps

- **The private key should not be lost from**
  - Expiration (recovery of tape allowed with expired certificate)
  - Revocation (unless backup is available for data)
  - Not all keys by algorithm (RSA, Standard support) are equal
    - Some sites may have limitations on key size or
    - Use different configuration setup
- **Beware storing RSA keys using the hwkeytool option of RETAIN**
  - This causes the RSA keys being created to ONLY be stored within the secure hardware crypto CEX2C and no backup copy is available
  - Those keys are lost with replacement of feature

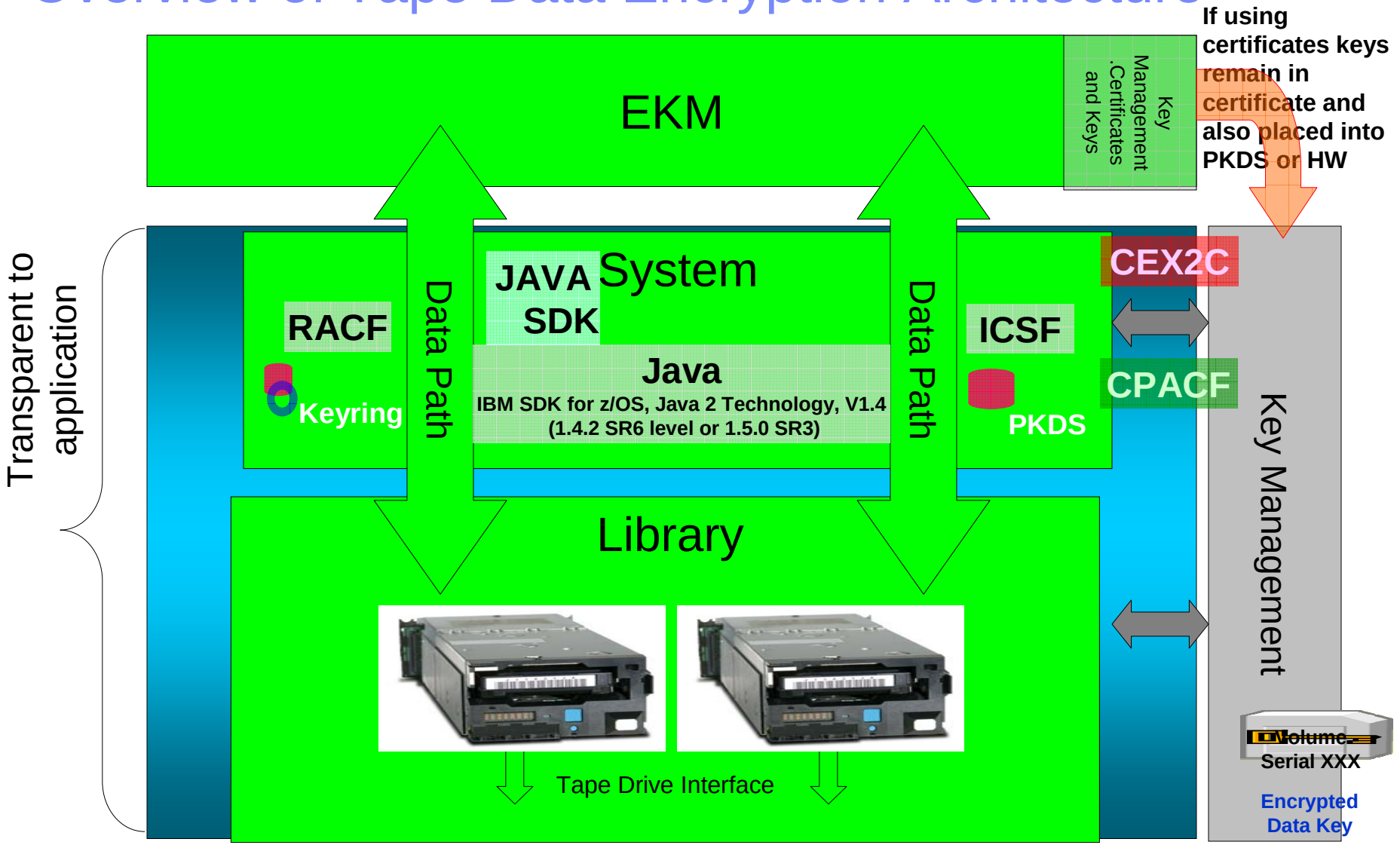
Perils and Hidden Traps . . .

- **Consider designing a naming convention for the tape encryption key labels to avoid**
  - Duplicate labels (not allowed)
  - Not recognizing keys protecting highly important, valuable data and inadvertently deleting the keys or changing their values
  - Not knowing when keys protecting important archived data having legal requirements for available access for a period of time that may be longer than expiration dates
- **Encryption or any cryptographic function**
  - Has ramifications caused by multiple conditions that impact decisions and/or outcome
  - Require an understanding of the technology, standards, and the various providers, implementations, etc.

Perils and Hidden Traps . . .

- **If PKDS is used, Master Keys are required**
  - Yes - another level of administration and work
  - Should not overlook the training required to understand each component
  - Any loss of Asymmetric Master Key requires reentry of SAME key values again
  - DR site will need to have copies of the same Master Key parts to enter at the DR site
- **Actual crypto usage can be requested in a variety of application callable services**
  - Some services are requested but will not necessarily use the crypto hardware
  - Other system related variables can cause problems that appear as crypto hardware performance problems

Overview of Tape Data Encryption Architecture



zSeries & z/OS IBM JCE Providers Overview

Keystores	Provider	Java Level	TS1120	Only LTO 4
JCE4758KS JCECCAKS	IBMJCE4758 IBMJCECCA	SDK 1.4.2 SDK 5.0		**Yes** **Yes**
JCE4758RACFKS JCECCARACFKS	IBMJCE4758 IBMJCECCA	SDK 1.4.2 SDK 5.0	**Yes** **yes**	
JCERACFKS			**Yes**	
JCEKS	IBMJCE		**yes**	**Yes**

zSeries & z/OS IBMJCE Provider Keystore Overview . . .

- **Keystores supported by IBMJCE provider**
  - JCEKS
  - JCERACFKS

- **JCEKS KeyStore for Certificates with the JCEKS provider**
  - Java-based keyring supported on all EKM platforms where EKM runs
  - JCEKS provides password-based protection for the contents of keystore for security
  - Relatively easy to copy contents of keystore for backup and recovery
  - Relatively easy to keep EKMs synchronized for failover

zSeries & z/OS IBMJCE Provider KeyStore Overview. . .

- **JCERACFKS KeyStore for Certificates with JCEKS provider**
  - SAF keyring keystore supported only on z/OS
  - Keystore created & managed through RACDCERT or equivalent
  - No ICSF is involved and software key storage is implied
  - SAF keyring is a collection of certificates that identify a trust policy

Note: KS = key store, CCA = Common Crypto Architecture 4758 = a CCA hardware device RACF = Resource Access Control Facility

zSeries & z/OS IBMJCE Provider KeyStore Overview . . .

- **JCERACFKS KeyStore for Certificates**
  - Only supported on z/OS platform
  - System Authorization Facility (SAF) keyring where certificates are stored after creation
  - Store certificate and key material within a SAF provider
  - Certificates stored in RACF keyring have a public key and may have a private key if the PCICC parameter has not been used
  - Symmetric keys are not supported so cannot use with LTO 4 tape drives

Note: KS = key store, CCA = Common Crypto Architecture 4758 = a CCA hardware device RACF = Resource Access Control Facility

zSeries & z/OS IBMJCE Provider KeyStore Overview . . .

- **JCERACFKS** (Certificates and key material stored in RACF)
  - SAF keyring-based keystore supported on z/OS only.
  - Keystore can be created and managed through the **RACDCERT** or equivalent SAF command
  - This keystore uses certificates generated in RACF/SAF and the key material is stored in RACF/SAF.

What You Need to Consider

- **Placement of recommended second EKM**
- **Platform differences will affect Provider and Keystore selections**
- **Consider what will be available at the disaster recovery site**
- **Consider where the data resides that must be protected by encryption**
- **Knowledge of Java**
- **Level of Java installed**

What You Need to Consider

- **Key strength**
- **Whether fewer algorithms are acceptable within your enterprise**
- **What role performance plays balanced against security level**
- **What type of tapes are involved; TS1120, LTO 4, a mixture?**
- **If exchanging tapes with a partner will you synch key label or use public key hash for encoding**
- **Archive requirements for certificate expiry or at least private key life**

Publications

- *DFSMS Software Support for IBM TotalStorage Enterprise Tape Drive TS1120 (3592) - SC26-7514-03*
- *Refer to IBM Encryption Key Manager component for the Java platform Installation, Planning, and User's Guide, GA76-0418, for details.*
- *keytool - Key and Certificate Management Tool http://www-128.ibm.com/developerworks/java/jdk/security/142/secguides/keytoolDocs/KeyToolUserGuide-142.html*
- *hwkeytool - Key and Certificate Management Tool http://www-03.ibm.com/servers/eserver/zseries/software/java/hwkeytool.html*
- *Java ™ Cryptography Architecture (JCA) Reference Guide download.java.net/jdk6/docs/technotes/guides/security/crypto/CryptoSpec.html*

Publications . . .

- *JavaTM Cryptography Extension (JCE) in J2SE API Specification & Reference http://www-03.ibm.com/servers/eserver/zseries/software/java/API_users_guide.html*
- *Java™ 2 on z/OS http://www-03.ibm.com/servers/eserver/zseries/software/java*
- *Java White Paper (Basics) http://java.sun.com/docs/white/langenv/*
- *JCE Hardware Cryptography IBMJCECCA Overview http://www-03.ibm.com/servers/eserver/zseries/software/java/j5jcecca.html*
- ***z/OS Unique Considerations for the JavaTM 2 SDK, Standard Edition, v 5.0***

  ftp://ftp.software.ibm.com/s390/java/zOSHWCryptoRefGuide.html