Chapter 16. Implementing Hardware-Based Tape Encryption

Note!

This chapter is a replacement for "Chapter 16. Implementing Hardware-Based Tape Encryption" in the *z/VSE Administration*, SC33-8304-00 (March 2007). It is for use by customers who wish to take advantage of the latest tape encryption functionality. These are the required APARs you must install:

- z/VSE 4.1 customers must install DY46682.
- z/VSE 3.1.x customers must install DY46685, PK43473, and PK43465.

This chapter describes how you can encrypt tapes using the hardware-based encryption facilities provided by an *encryption-capable* tape device. An example of an encryption-capable tape device is the IBM TotalStorage 3592 Model E05[2].

This chapter contains these main sections:

Related Information:

For details of the ...	Refer to the ...
• Query Tape (QT) command, which you use to display the *key encrypted key labels* of the tape cartridge • JCL ASSGN statement, which you use to specify the mode for an encryption-capable tape device • JCL KEKL statement, which you use to force encryption when writing data to a tape	chapter "Job Control and Attention Routine" in the *z/VSE System Control Statements*, SC33-8305.
LIBR routine used for backing up encrypted z/VSE libraries, sub-libraries, and members to tape	chapter "Librarian" in the *z/VSE System Control Statements*, SC33-8305.
current encryption-capable tape devices and tape controllers	chapter "Hardware Support" in the *z/VSE Planning*, SC33-8301.

2. The IBM TotalStorage 3592 Model E05 has been renamed to the IBM System Storage TS1120.

Overview of Hardware-Based Tape Encryption

Figure 127 provides a simplified description of hardware-based tape encryption.

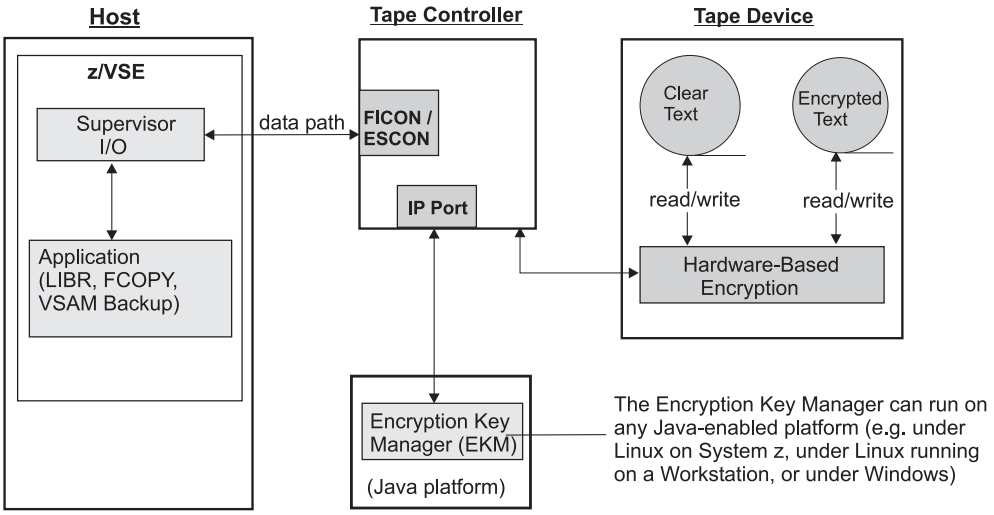

Figure 127. Overview of Hardware-Based Tape Encryption

Here is an explanation of Figure 127:

1. A request is made for data to be backed-up to tape in *encrypted format*. This request can originate from a:
   - LIBR, FCOPY, or VSAM backup job running on the z/VSE host.

   An ASSGN statement (contained in the job or entered at the console) sets the Mode for the tape unit to 03, 0B, 2B, or 23 (all of which require encryption).

   A KEKL statement (contained in the job or entered at the console) containing one or two key-encryption-key labels informs z/VSE to associate a tape unit with one or two key-encryption-key labels.
2. The key-encryption-key labels are passed via the z/VSE Supervisor to the Encryption Key Manager (EKM).
3. "Key negotiation" takes place between the tape device and the EKM, during which the EKM validates/supplies encryption keys with the tape device. The tape device and EKM communicate via the TCP/IP protocol.
4. If the key-verification process is successful, the data on the tape cartridge will be encrypted. If not, an error message is returned.

Prerequisites for Using Hardware-Based Tape Encryption

These are the prerequisites for using hardware-based tape encryption:

- The Encryption Key Manager must be running on a Java platform. For details, see "Obtaining and Installing the Encryption Key Manager" on page 493.
- You must have installed and configured an encryption-capable tape device and tape controller. For a list of currently-supported tape devices, refer to the chapter "Hardware Support" in the *z/VSE Planning*, SC33-8301
- You must record your data using the *Encrypted Enterprise Format 2* (EEFMT2), which is the encrypted form of EFMT2. The EEFMT2 recording format is supported across all of the 3592 media types (MEDIA5 to MEDIA10). For details, refer to the chapter "Hardware Support" in the manual *z/VSE Planning*, SC33-8301.

Restrictions When Using Hardware-Based Tape Encryption

The following restrictions apply to the use of encrypted tapes:

- A tape cartridge cannot contain both encrypted and non-encrypted data.
- If the first file written to a tape is encrypted, all subsequent files written to that same tape cartridge will be encrypted using the *same key* (except for the volume label structure for the first file sequence).
- The DOSVSDMP utility is *not supported for encryption*. Therefore, a request to create an encrypted standalone dump tape using the DOSVSDMP utility will be rejected!

Obtaining and Installing the Encryption Key Manager

The Encryption Key Manager (EKM) is a common-platform Java application that is used to generate and protect AES (Advanced Encryption Standard) keys. Upon request, the EKM generates AES keys to be used for encryption, and protects these keys using RSA key pairs.

To obtain a copy of the EKM from the Internet, you should:

1. Enter the following URL:

   ```
   http://www.ibm.com/support/us/
   ```

2. Search for "Encryption Key Manager" and locate the zipped file containing the EKM.
3. Download the zipped file containing the EKM to the directory where you want to install it.
4. Install and customize the EKM by following the instructions provided in the *IBM System Storage Tape Enterprise Key Manager, Introduction, Planning and User Guide*, GA76-0418.

Using a Job to Backup Data With Encryption

Jobs for backing-up to tape with encryption (LIBR, FCOPY, VSAM Backup) *must* contain an ASSGN statement and *might* contain a KEKL statement. An example for a LIBR job is given below.

Example of a LIBR Job to Backup/Encrypt the Contents of a Library

The LIBR job below will backup to tape and encrypt the contents of library PRD2. It specifies *two* key-encryption-key labels (KEKL1 and KEKL2). The tape device has a unit address (cuu) of 480.

```
// JOB ENCRYPT
// ID USER=user-ID,PWD=password
// ASSGN SYS005,480,03
// KEKL UNIT=480,KEKL1='HUSKEKL1',KEM1=L,KEKL2='HUSKEKL2',KEM2=L
// EXEC LIBR
BACKUP LIB=PRD2 TAPE=SYS005
/*
/&
```

Specifying KEKL Statements

Jobs for backing-up to tape with encryption (LIBR, FCOPY, VSAM backup) *might include* a KEKL statement.

If your job does *not contain* a KEKL statement, the EKM will *use the defaults* that you previously generated and stored in the EKM.

The KEKL statement has the following syntax:

```
// KEKL UNIT={cuu|SYSnnn},KEKL1='kekl1',KEM1={L|H},KEKL2='kekl2',KEM2={L|H}
// KEKL UNIT={cuu|SYSnnn},KEKL1='kekl1',KEM1={L|H}
// KEKL UNIT={cuu|SYSnnn},CLEAR
```

where:

cuu Specifies the tape unit for which the key-encryption-key labels are to be used.

SYS*nnn*
Specifies the logical unit of the tape unit for which the key-encryption-key labels are to be used. *This logical unit must have been previously assigned*. The value of *nnn* can be:
- between 000 and 255
- LST
- PUN

kekl1 Is the label for the first key-encryption-key to be used by the EKM to encrypt the data encryption key. Must be enclosed in single quotation marks.

kekl2 Is the label for the second key-encryption-key to be used by the EKM to encrypt the data encryption key. Must be enclosed in single quotation marks.

L|H Specifies the "encoding mechanism" (KEM). The KEM specifies how the labels for the first key-encryption-key (KEKL1) and second key-encryption-key (KEKL2) is encoded by the EKM and stored on the tape cartridge. The values can be either:
L Encoded as the specified label.
H Encoded as a hash of the public key.

CLEAR
Indicates that the information previously established by a KEKL statement is cleared.

Note: You might need to reset the KEKL (the default KEKL, or the KEKL from a previous KEKL statement) on a previously-encrypted volume. To do so, you must issue a WRITE command (for example, writing a tape mark) from the Beginning-Of-the-Tape (BOT) with *encryption mode not active*.

For further details about using the KEKL statement, refer to the chapter "Job Control and Attention Routine" in the *z/VSE System Control Statements*, SC33-8305.

Specifying ASSGN Statements

Jobs (LIBR, FCOPY, VSAM backup) that require hardware-based tape encryption **must** include an ASSGN statement as follows:

Method 1: Specify the Device Mode of the Encryption-Capable Tape Device.

The syntax of the ASSGN statement is as follows:

```
// ASSGN SYSnnn,cuu,mode
```

where:

- *cuu* is the device address of the encryption-capable tape device.
- *mode* is a 1-byte field that determines how the data on the tape should be written. These are the encryption-related modes you can use:
  X'03' Encryption Write Mode
  X'0B' Encryption and IDRC (compression) Write Mode
  X'23' Encryption with unbuffered Write Mode
  X'2B' Encryption and IDRC (compression) and unbuffered Write Mode

Note: IDRC is an abbreviation for Improved Data Recording Capability.

Method 2: Let z/VSE Find a Suitable Tape Device

Here, you specify that you require an encrypted write-format, and then let z/VSE locate a suitable tape device. The syntax of the ASSGN statement is as follows:

```
// ASSGN SYSnnn,device_class,mode
```

If the tape *device_class* is set to EEFMT2, z/VSE will search your system for an encryption-capable tape device. The *mode* specifies any encryption mode.

For further details about using the ASSGN statement, refer to the chapter "Job Control and Attention Routine" in the *z/VSE System Control Statements*, SC33-8305.

Using the Query Tape (QT) Command to Display Tape Information

You can use the Attention Routine (AR) **QT** command to display the mode of a tape device that is attached to your z/VSE system.

In the following example:

- CODE 5603 indicates:
  – A tape device that uses the TPA is attached to z/VSE (the **56** part of 5603).
  – This tape device is assigned to encryption mode (the **03** part of 5603).
- 3592-E05 is the device type for the IBM TotalStorage 3592 Model E05 tape device.
- KEY_LABEL_001 is the label for the first key-encryption-key to be used by the EKM to encrypt the data encryption key.
- KEY_LABEL_002 is the label for the second key-encryption-key to be used by the EKM to encrypt the data encryption key.

```
QT A83
AR 0015 CUU  CODE DEV.-TYP   VOLID  USAGE   MED-TYP   STATUS    POSITION
AR 0015 A83  5603 3592-E05   PAUL01 BG      CST5 /E   RESERVED  8 BLK
AR 0015      CU   3592-C06           LIB     3494-L10  (GALL88)
AR 0015                              FAST-ACC.SEG.=     0 MB    FILES = 2
AR 0015 KEKL1:KEY_LABEL_001
AR 0015 KEKL2:KEY_LABEL_002
AR 0015 1I40I  READY
```

Figure 128. Using the QT Command to Display the Details of an Encrypted Tape

For further details, refer to the chapter "Job Control and Attention Routine" in the *z/VSE System Control Statements*, SC33-8305.

Using the LIBR Utility to Read the Contents of an Encrypted Tape

Figure 129 is an example of how to read the contents of an encrypted tape using the LIBR utility:

- This job does *not* require KEKL statements to specify the encryption keys to be used.
- If KEKL statements are included in the job, they will be ignored.
- To read the encrypted data, the job uses the keys that are already stored on the tape.

However, these are the prerequisites for running this job:

- The z/VSE host where the job is to run must be connected to an EKM.
- The tape must have been previously encrypted using keys that are known by the currently-connected EKM.
- The encryption keys must not have been deleted from the currently-connected EKM.

```
* $$ JOB JNM=LIBSCAN,DISP=D,PRI=3,                                    C
* $$ NTFY=YES,                                                        C
* $$ LDEST=*,                                                         C
* $$ CLASS=0
// JOB LIBSCAN  SCAN VSE LIBRARY BACKUP TAPE
* THIS FUNCTION USES A TAPE FOR INPUT
* MOUNT TAPE BACKUP ON DEVICE 480
* THEN CONTINUE. IF NOT POSSIBLE CANCEL THIS JOB.
// PAUSE
// MTC REW,480
// ASSGN SYS004,480
// EXEC LIBR,PARM='MSHP'
   RESTORE *                        /* LIBRARY IDENTIFICATION */ -
          SCAN  = YES               /* SCAN SPECIFICATION     */ -
          TAPE = SYS004             /* TAPEADDRESS            */
/*
// MTC RUN,480
/&
* $$ EOJ
```

Figure 129. Using a LIBR Job to Read the Contents of an Encrypted Tape

Understanding Message 0P68I KEYXCHG ER

The EKM generates this message explanation:

```
0P68I Encryption key negotiation with the EKM failed
```

However, the *sense data* of the message contains additional useful information. For example:

```
804C08C02240 2751 0001FF0000000000 0005EE3100000092 2004E82061BA2111

CU=00 DRIVE=000000 EKM=EE31
```

In the above example (which starts at byte 0), bytes 4 and 5 might contains **2240**. This is the return code and reason-qualifier code (RC-RQC). It means that the required encryption-key exchange *has failed*. Furthermore:

1. Byte 8 contains the *CU reason code* (in the above example, **00**).
2. Bytes 13 ,14 and 15 contain the *sense key from the device* (in the above example, **000000**).
3. Bytes 18 and 19 contain the *sense key from the EKM*. The value **EE31** means that an encryption configuration problem has occurred in which the error has something to do with the *key store*.

For further details of this and other EKM error messages, refer to the *IBM System Storage Tape Enterprise Key Manager, Introduction Planning and User Guide*, GA76-0418.

Hints and Tips

This section contains various hints and tips that you might find useful.

Assigning System Logical Units

This problem-situation might arise:

1. The assignment of system logical units results in OPEN processing (during VOL1 and header-label checking). After OPEN processing:
   a. The tape (for labeled tapes) is positioned *behind* the VOL1 label.
   b. Previously-used KEKLs are still active.
2. A subsequent KEKL statement that is set *after* the ASSGN statement causes the job *to be cancelled*.

To overcome this problem, you can create the following job:

```
// ASSGN SYSnnn,cuu,mode
// MTC REW,cuu
// KEKL UNIT=cuu,KEKL1='TEST',KEM1=L
```

The *mode* must be one of the encryption modes (for example, **03**).

Positioning of the Tape When Using the ASSGN Statement

The syntax of the ASSGN statement is as follows:

```
ASSGN SYSnnn,cuu,mode
```

If the tape specified in the *cuu* device address is *at load point*, the new *mode* setting is *immediately effective*.

If the tape specified in the *cuu* device address is *not at load point*, the new *mode* setting will be effective *the next time a write occurs at load point*.

For *mode* set to encryption X'03':

- If the tape was *at load point*, the tape will be written *as encrypted*.
- If the tape was *not at load point*, the tape will continue writing *in the current mode*.

If the first file written to a tape is encrypted, all subsequent files written to that *same tape cartridge* will be encrypted using *the same data key*.

Handling Situations Where the EKM is not Available

If a tape contains encrypted data and is rewritten *without* encryption activated, the job might fail with a key-exchange error (described in "Understanding Message 0P68I KEYXCHG ER" on page 496).

This is because certain standalone utilities read the VOL1 label *before* starting the I/O process.

To overcome this problem, before you resubmit the job you should:

1. write a tape mark,
2. rewind the tape.

Running Standalone Utilities (FCOPY, ICKDSF, DITTO, LIBR)

The standalone utilities FCOPY, ICKDSF, DITTO, and LIBR can be called from an encrypted standalone backup tape.

Backups performed from any of these utilities will be in *unencrypted format* only.

Additional Considerations When Using LIBR Utility

A LIBR BACKUP job with RESTORE=STANDALONE can be written in encrypted format.

- If an IPL is made from an *unlabeled* tape, there might be a delay when the key-exchange occurs. You might be required to re-IPL the tape device using the `IPL cuu` command.
- If an IPL is made from a *labeled* tape, you might have to enter the `IPL cuu` command approximately *four times*, until the tape marks at the beginning of the tape have been skipped. This problem can also occur without encryption.

Overwriting Encrypted Volumes

If an encrypted volume is processed but the key is unknown to the EKM, access might fail with the message "0P68I Key Exchange Error" (described in "Understanding Message 0P68I KEYXCHG ER" on page 496).

To overcome this error, you can write a tape mark at the Beginning-Of-the-Tape (BOT).

Multivolume File Processing

To process a multivolume file on an *alternate* volume, you must specify the *same KEKL* as was specified for the *original volume*.

Here is an example of how to process a multivolume file. In this example, the specified alternate tape must also be assigned to encryption mode.

```
// ASSGN SYS005,cuu1,3
// ASSGN SYS006,cuu2,3
// ASSGN SYS005,cuu2,ALT
// KEKL UNIT=cuu1,KEKL1='TEST',KEM1=L
// KEKL UNIT=cuu2,KEKL1='TEST',KEM1=L
```