TS7700 & Encryption

G13

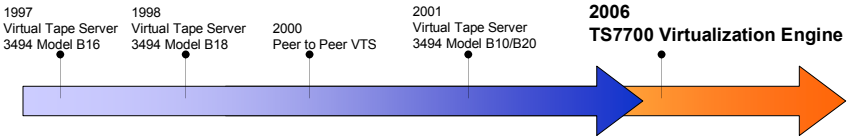
M.G. McCullough

Agenda

- TS7700 Overview
- Encryption Overview
- Courses available
- Questions

Ten Years of IBM Enterprise Tape Virtualization

- First Virtual Tape Server in 1997
- IBM 3494-B16
  - Capacity TVC 72 GB
  - Throughput 6 MB/s
- Generation 4 with new architecture in 2006
  - IBM TS7700 Virtualization Engine
    - Capacity TVC 18 TB (compression rate 3:1)
    - Throughput up to 600 MB/s



TS7700 Virtualization Engine - Specifications

Specification	TS7740		Model B10			Model B20		Model B18		
Number of Virtual Devices	256	512 [1]	64			128	256	64	128	
Usable Cache Capacity	2 TB - 6 TB		216 / 432 GB			864 GB to 1.7 TB		72 GB to 1.7 TB		
Compressed Cache Capacity (3:1)	18 TB		648 GB to 1.2 TB			2.4 TB to 5.2 TB		216 GB to 5.2 TB		
FICON	2	4	2	4		4	8			
ESCON			2	4	8	8	16	2	4	8
TS1120/3592 Tape Drive Attachment	4 - 16		4 - 12			4 - 12				
3590 Tape Drive Attachment			4 - 6			4 - 12		3 - 12		
Number of virtual Volumes	500,000		250,000			500,000		250,000		
Conversion Upgrade	planned					planned				

Statements of IBM future plans and directions are provided for information purposes only. Plans and direction are subject to change without notice.

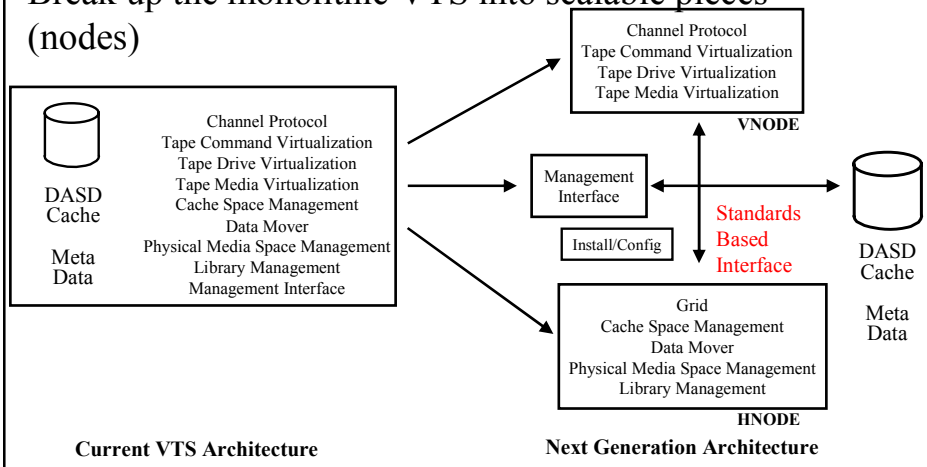
[1] in a Grid configuration

TS7740 Architecture

- Separates the functionality of the system into smaller components with well defined, open interfaces
- Provide a platform which the smaller components can be tailored from a small solution into an extremely large one
  - Grows with the customer's needs
- Allows the customer to plug in different components of the same functionality, to provide a solution for specific customer environment
- Every configuration has multi-site logic inherent to its operation

TS7700 - Architectural Partitioning

- Break up the monolithic VTS into scalable pieces (nodes)

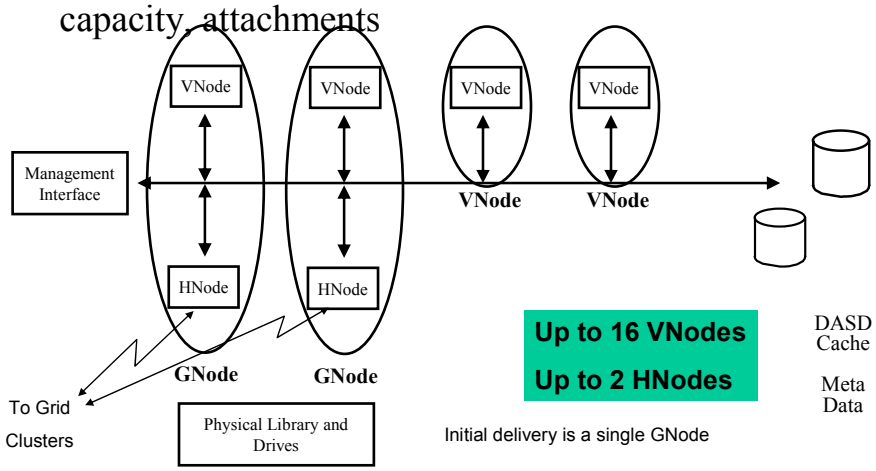


Current VTS Architecture

Next Generation Architecture

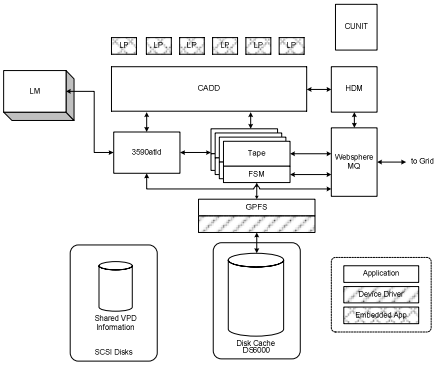
TS7700 - Scalability

•Common DASD, Multiple Elements - redundancy, capacity, attachments



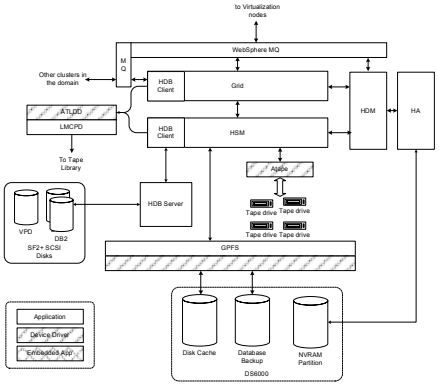
Virtualization Node (VNode)

- The "VNode" refers to a code stack which performs all of the actions needed to present a library image and drive images to a host
- The VNode code was designed to run along side of the HNode code in the same controller, or in a separate controller
- Uses standardized interfaces to talk with outside components (TCP/IP)



Hierarchical Storage Management Node (HNode)

- The "HNode" refers to a code stack which performs all of the actions needed coordinate the contents of the disk cache with the data on backend tape. It also includes the logic for managing changes and replication of the data across different sites.
- The HNode code was designed to run along side of the VNode code in the same controller, or in a separate controller
- Uses standardized interfaces to talk with outside components (TCP/IP)



General Node (GNode)

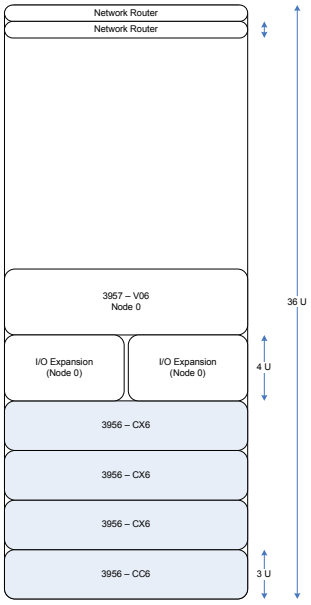
- A GNode can be considered a VNode and an HNode sharing the same physical controller
- A GNode can be considered the equivalent of today's VTS, such that a single controller has both the VNode and HNode capabilities

TS7700 Components

- The TS7700 is comprised of one or more sites containing a cluster
- A cluster is a complete Virtual Tape Subsystem
  - A Virtualization Engine Node (3957-V06)
  - A Virtualization Engine Cache Controller (3956-CC6)
  - Virtualization Engine Cache Drawers (3956-CX6)
  - A library manager controller (3953)
  - A physical tape library (3584) and tape drives (3592)
  - A frame to house the controller and disk (3952-F05)
  - Other infrastructure components
    - Ethernet Routers and switches
    - Fibre channel switches

TS7740 Frame Layout

- pSeries Server
  - Power5 processor (2 dual core CPUs)
- I/O Expansion drawers for pSeries
- RAID Disk controller and 3 disk expansion drawers
- Redundant network routers
  - Provides internal network connection to Library Manager and Disk Controller configuration
  - Provides protected NAT interface for customer to access Management Interface services running on controller
- Expansion for 2nd 3957 controller and I/O drawers



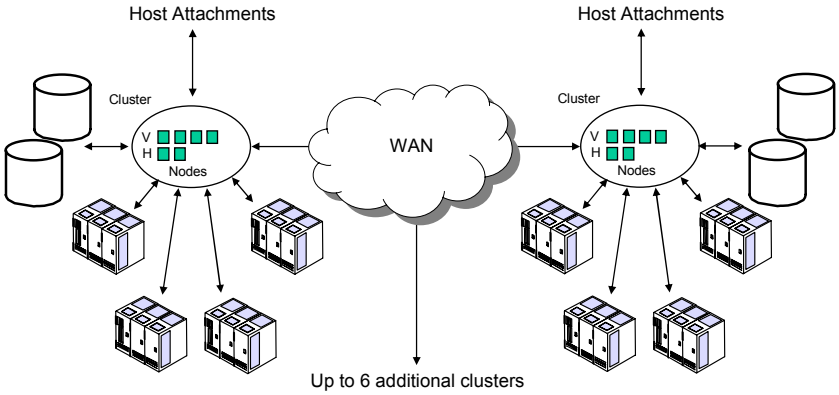
TS7740 Grid – The Evolution of Peer-to-Peer

- Multiple clusters can be connected to one another using a TCP/IP WAN, to form a Grid
  - Interconnections between the sites uses standard TCP/IP Networking infrastructure
- Every TS7740 configuration has multi-site capabilities and logic built into it (even single site configurations)
  - Token database is incorporated into every GNode/HNode
  - Each HNode/GNode contains a "Grid" code layers which tracks updates to all logical volumes (whether there is only one site, or multiple sites)
- Any logical volume can be mounted and accessed from any virtual device in the subsystem
  - Some mounts of logical volumes will be accessed from the site containing the consistent copy of the logical volume (TVC selection)

TS7740 Grid – The Evolution of Peer-to-Peer

- Every node (HNode, VNode, GNode), has two network adapters for cross-site communication
  - VNodes use link for Remote File access
  - HNodes use link for site-site messages and replication of logical volumes
  - Scaling the number of nodes increases the capabilities for moving data between the sites

TS7740 - Scalable Architecture



Architecture allows scaling from 128-32768 virtual drives, .5 to 4 million virtual volumes, 1-100's TB of cache, 1-32 physical libraries and 4-1024 physical drives, single and multi-site business continuity

Site-Site Replication Policy (2)

- Consistency Policy Options for each site
  - No copy (N) – this site does not receive a copy for volumes in this management class
  - RUN (R)– this site will have a valid replication of the logical volume before we provide Device End to the Rewind Unload (RUN) command from the host (this is a direct parallel to current PtP Immediate mode copy setting)
  - Deferred (D) – this site will get a valid replication of the logical volume at some point in time after the job completes (same as the deferred mode in the PTP)

Site-Site Replication Policy (1)

- New definition for copy as Copy Consistency Point
  - Defines when a target volume is consistent on a TS7740 with the source volume
- The Consistency Policies determine:
  - The site which should have the initial location of data (TVC selection)
  - Which sites get copies of logical volumes, and at what point the site should have copies
- Consistency policies are configured through the Management Class
  - Policies are assigned to a management class name
  - Policies are set through Library Manager

Site-Site Replication Policy (3)

- Setting of the policy at the LM defines the consistency sync point for every defined site in the subsystem
  - The policies become an array of sync values, with each element of the array representing the policy for a given site
  - If it is desired to have a copy of a volume at unload time at site A, and a deferred copy at site B, the array would be:

Site A: | R | D | Site B: | R | D |

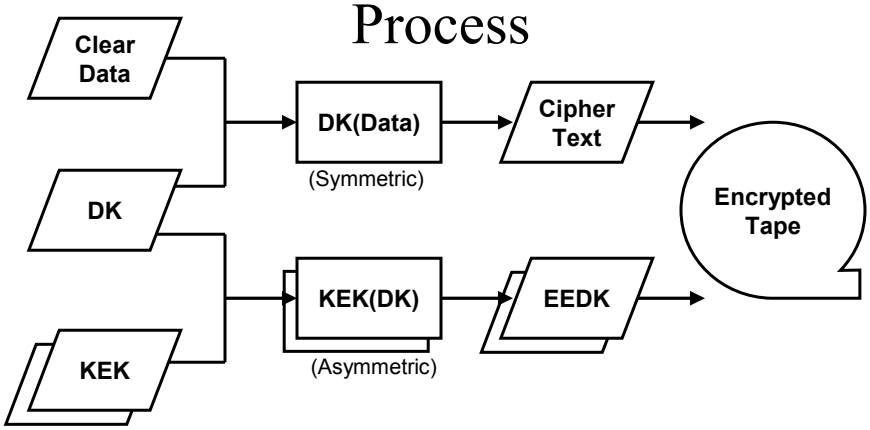
- The replication policies can be different at each LM
  - The setting of the policies at each LM dictates what the resulting actions would be if the volume is mounted for a virtual device address associated with the LM of that site.
  - If a volume is mounted to a device of site A, they wish to have a valid copy at site A at RUN, and deferred at site B. If the volume is mounted to a device at site B, they wish to have a valid copy at RUN at B, and deferred to A. The arrays would be:

Site A: | R | D | Site B: | D | R |

TS7700 Course

- SS270 2 day class

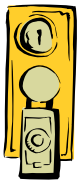
Encryption Review : Encryption Process



DK – Data Key (Symmetric)
KEK – Key Encrypting Key (Asymmetric)
EEDK – Externally Encrypted Data Key

Encryption Review : Wrapped Keys

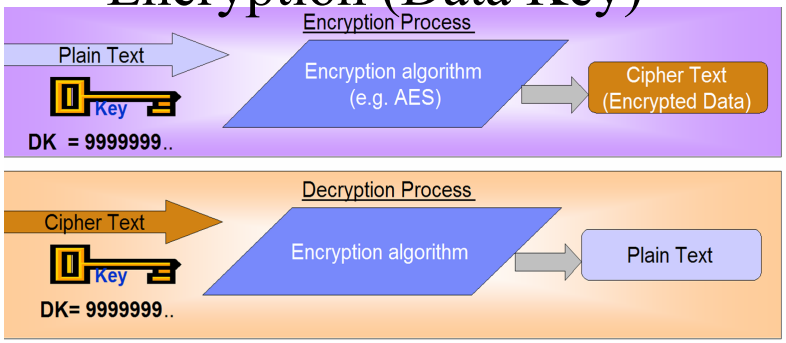
- Data Key (DK) – 00100111001000…
  - Symmetric Encryption AES-256
  - Random number generated by Crypto Provider Services
  - Used to encrypt/decrypt data on tape
  - Very fast



- Key Encrypting Key (KEK) Pair
  - Asymmetric Encryption RSA-2048
  - Created by the Customer/Business Partner/Third Party Provider
  - Public half used to encrypt DK (into EEDK)
  - Private half used to decrypt DK (from EEDK)
  - Referenced by KEK Labels (AKA Key Labels) or hashes

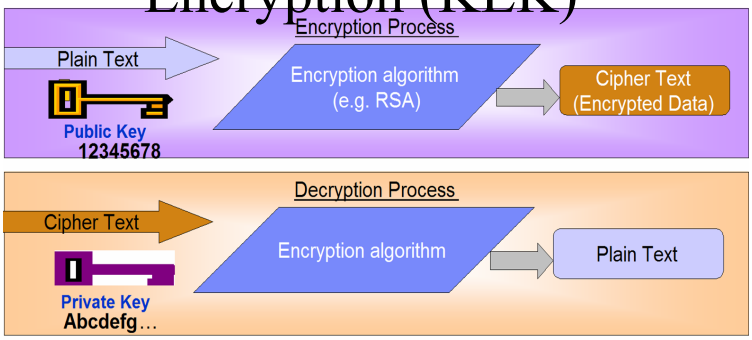
- Drive Session Key (dSK)
  - Asymmetric
  - Created by drive on a per-mount basis
  - Public half used to encrypt DK (into Session-Encrypted Data Key [SEDK])
  - Private half used to decrypt DK (from SEDK)

Encryption Review : Symmetric Encryption (Data Key)



- Data that is not encrypted – clear text
- Clear text is encrypted by processing with a “key” and an encryption algorithm
- Keys are bit streams, 256 bits for IBM drives
- Symmetric encryption – same key to encrypt and decrypt

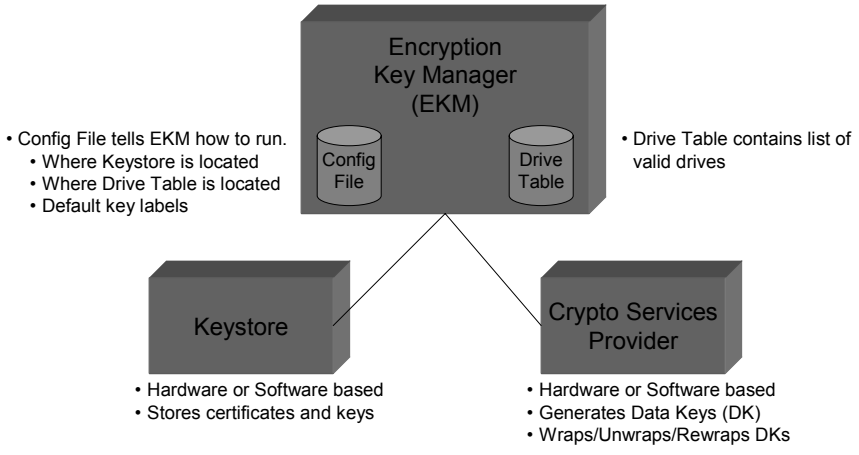
Encryption Review : Asymmetric Encryption (KEK)

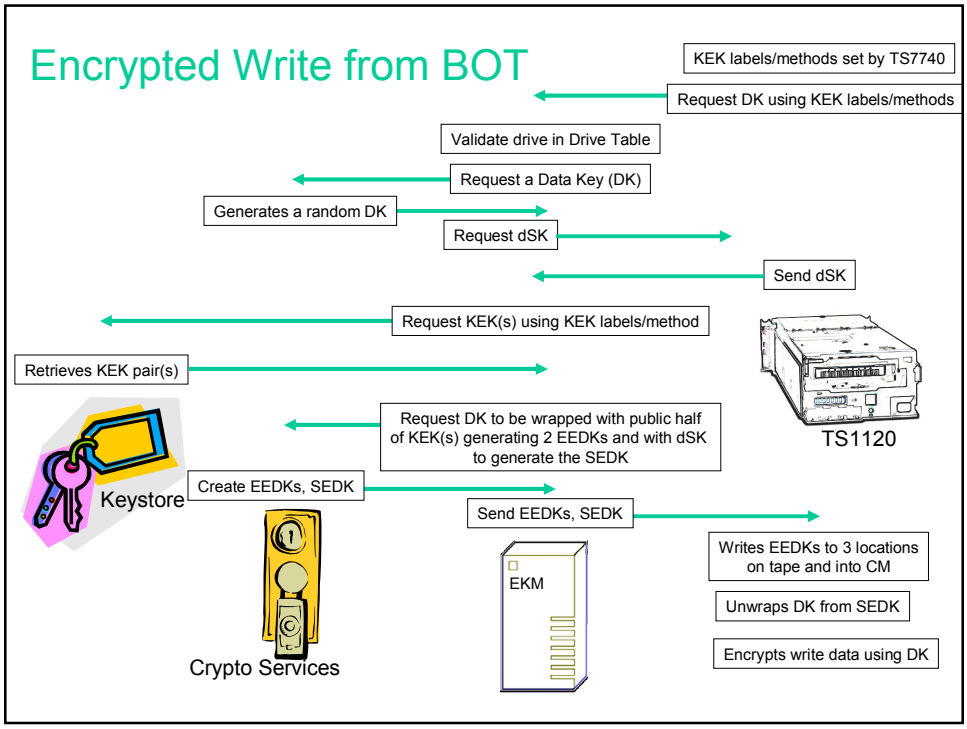


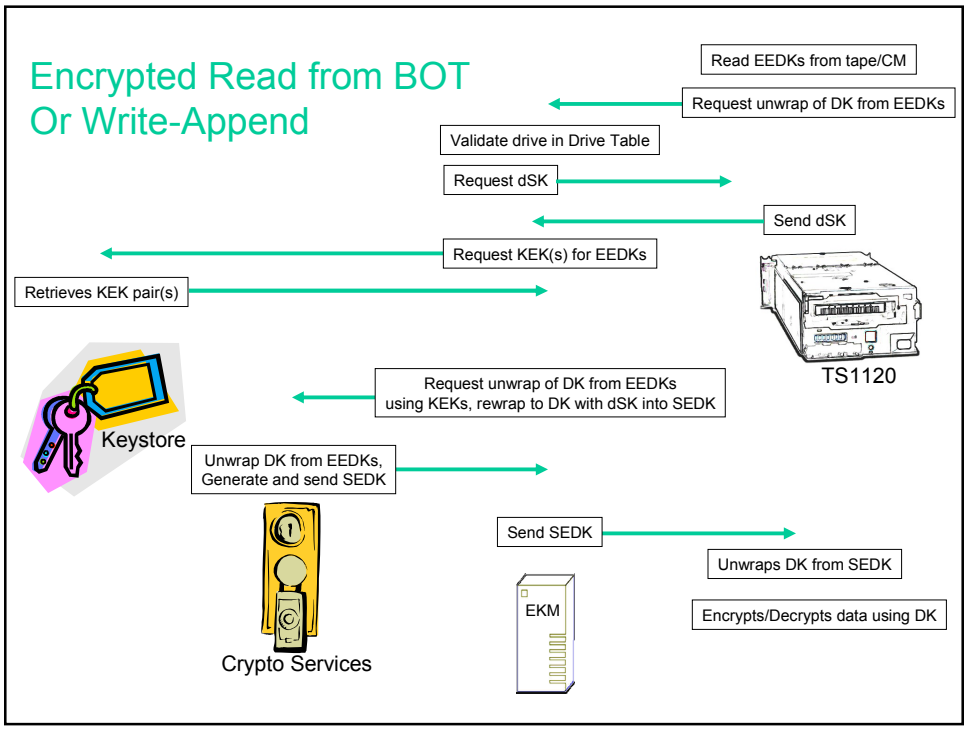
- The key used to encrypt is often referred to as the Public key, eg. the KEKs used to wrap the DK and create the EEDKs.
- The Public key may be made widely available without fear of compromise.
- The Key used to decrypt is referred to as the Private key.
- Private Keys must be secured against unauthorized access.
- Public / Private encryption is widely used for exchange of data between organizations.

Encryption Review : Encryption Key Manager

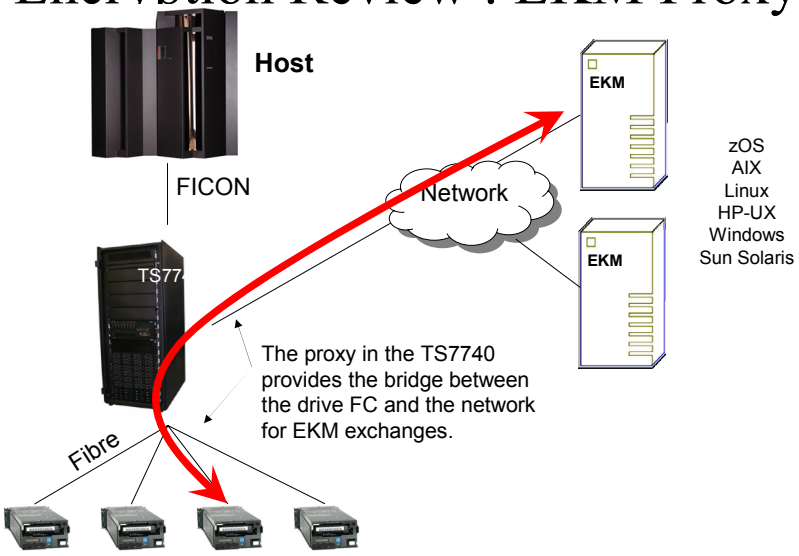
- Runs in IBM Java Runtime Environment (JRE)
- Supplied free from IBM
- Does not perform any crypto operations itself



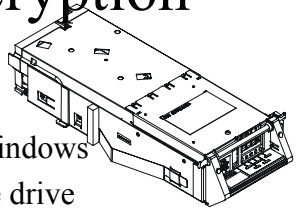

Encrypted Write from BOT
KEK labels/methods set by TS7740
Request DK using KEK labels/methods
Validate drive in Drive Table
Request a Data Key (DK)
Generates a random DK
Request dSK
Send dSK
Request KEK(s) using KEK labels/method
Retrieves KEK pair(s)
TS1120
Request DK to be wrapped with public half of KEK(s) generating 2 EEDKs and with dSK to generate the SEDK
Keystore
Create EEDKs, SEDK
Send EEDKs, SEDK
Writes EEDKs to 3 locations on tape and into CM
EKM
Unwraps DK from SEDK
Encrypts write data using DK
Crypto Services


Encrypted Read from BOT
Or Write-Append
Read EEDKs from tape/CM
Request unwrap of DK from EEDKs
Validate drive in Drive Table
Request dSK
Send dSK
Request KEK(s) for EEDKs
Retrieves KEK pair(s)
TS1120
Request unwrap of DK from EEDKs using KEKs, rewrap to DK with dSK into SEDK
Keystore
Unwrap DK from EEDKs, Generate and send SEDK
Send SEDK
Unwraps DK from SEDK
EKM
Encrypts/Decrypts data using DK
Crypto Services

Encryption Review : EKM Proxy



IBM Tape Data Encryption



- IBM Encryption Key Manager (EKM)
  - IBM Java component
  - z/OS, i5/OS, AIX, HP, Sun, Linux and Windows
  - Generates and serves keys to TS1120 tape drive
  - Obtains encryption keys from the keystore
- TS1120 Tape Drive
  - Addresses tape data security concerns
  - Standard feature on all new TS1120 Tape Drives
  - Chargeable upgrade feature for existing TS1120s

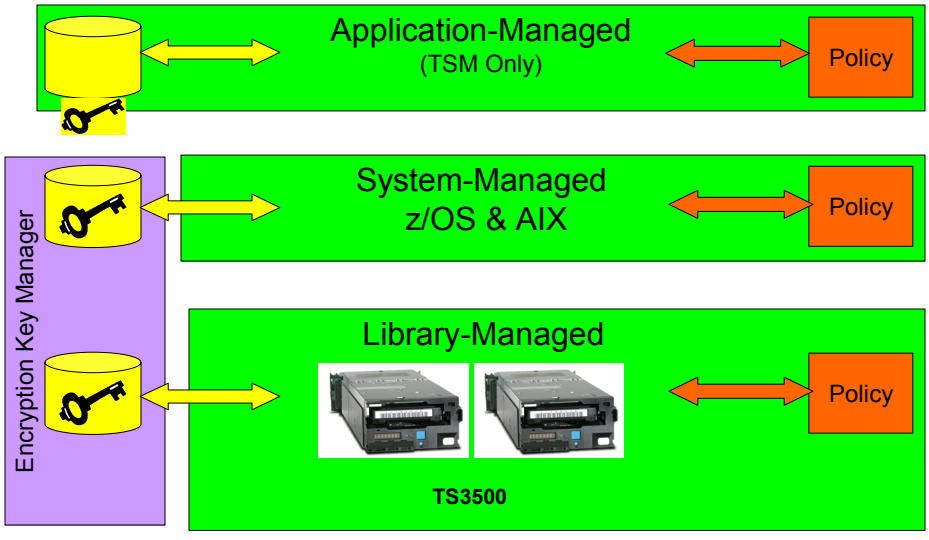
Encryption Key Manager

How to obtain the (EKM)

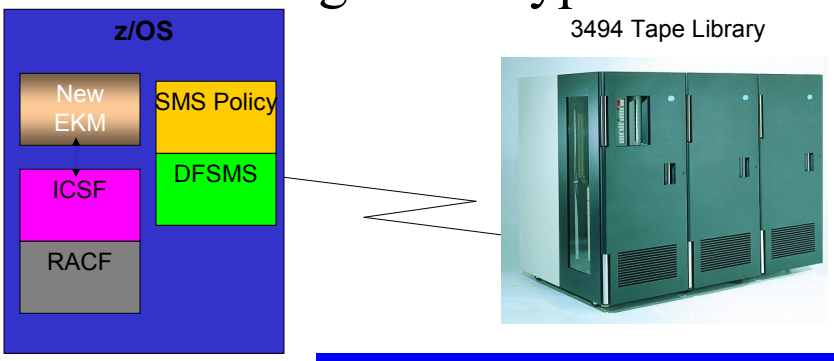
- Running EKM on AIX ® (included in base operating systems) – AIX 5.2 and above
  - One of the two may be used
    - Java SDK 5 (32 and 64 bit)
    - Java SDK 1.4.2 (32 and 64 bit)
  - Updates to the AIX Java SDK may be obtained at the following web site:
  - http://www.ibm.com/developerworks/java/jdk/aix/index.html
- Running EKM on Linux (included in base operating systems) –
- Running EKM on z/OS® – supported z/OS levels (1.6 & 1.7 10/2006)
  - IBM SDK for z/OS, Java 2 Technology Edition, V1.4, 5655-I56 (at the SDK1.4.2 SR6 or above level)
  - IBM 31-bit SDK for z/OS, Java 2 Technology Edition, V5.0, 5655-I98 (at the SDK5 SR3 or above level)
  - System p™, System x™, Sytem z™ (FCP only on z)
  - SUSE Linux Enterprise Server 9 (SLES 9) or Red Hat Enterprise Linux (RHEL 4)
  - One of the two may be used
    - Java 2 Standard Edition SDK 5
    - Java 2 Standard Edition SDK 1.4.2
  - Updates to the Linux Java SDK may be obtained at the following web site:
  - http://www-128.ibm.com/developerworks/java/jdk/linux/download.html
- Running EKM on i5/OS ® – i5/OS 5.2 and later
  - • IBM Developer Kit for Java - Java Developer Kit 5.0 - 5722-JV1
- Running EKM on HP, Sun, Windows – Available 12/1/2006

Encryption Key Manager available at no additional charge

Encryption Methods



Example – Single z/OS System Managed Encryption



Encryption enablement provided transparently to the application through DFSMS (Data Class)

Key management exchanges flow over

Feature Overview: Key Labels & Methods

- A "Clear Label" (KEK Label) is a symbolic reference to a Key-Encrypting Key (certificate in the key store).
- A "Hash Label" (KEK Hash) is generated directly from the KEK in the key store.
- You may specify keys to use by KEK Label only, but the references stored on tape may be by KEK Label or Hash.
- Different locations/EKMs may have different labels for the same KEK, but the hash will always match.
- Hash is important for interchange/DR, but Labels are descriptive.

Encryption Class

- SS270 1 ½ day class

Thank You

- Questions?