To: INCITS Technical Committee T10
From: Gideon Avida, Decru
Date: Jan. 17, 2007
Document: T10/06-389r5
Subject: Using Public-Key Cryptography for Key Wrapping

1 Revision History

Revision 0 (06-389r0): Posted to the T10 web site on August 25, 2006.
Proposal overview for discussion in the Nashua meeting.

Revision 1 (06-389r1): Posted to the T10 web site October 13, 2006.
Added threat model discussion (section 4.4).
Detailed proposal (added section 5).
Fixed typos and added more references.

Revision 2 (06-389r2): Posted to the T10 web site Nov. 10, 2006.
Revised as requested in the Nov. 7, 2006 SSC-3 meeting.
Changed ECC-521 public key representation to uncompressed.

Revision 3 (06-389r3): Posted to the T10 web site Dec. 7, 2006.
Incorporated comments from EMC.

Revision 4 (06-389r4): Posted to the T10 web site Dec 14, 2006
Added a section for addition into the model clause
Incorporated comments from EMC and the Dec. 11th, 2006, con-call.

Revision 5 (06-389r5): Posted to the T10 web site Jan. 17th, 2007
06-389r4 as modified in the Jan. 16, 2007 SSC-3 meeting

2 Introduction

Public-Key Cryptography enables sending secrets securely without requiring a prior shared secret.
There are several advantages for this approach:

1. There is no need to generate and maintain a secure channel and its related security association.
2. Keys can be generated/stored and wrapped away from the application client. This reduces key exposure as there may be multiple client servers (usually running a general purpose OS) and only a few key management servers.
3. There is flexibility in complexity vs. security.

The proposed algorithms are PKCS #1 v2.1 RSAES-OAEP and ECIES.

3 References

1. The IEEE P1363 Standard Specifications For Public-Key Cryptography — Amendment 1: Additional Techniques
http://grouper.ieee.org/groups/1363/tradPK/index.html
2. ANSI 9.63-2001
http://webstore.ansi.org/ansidocstore/subscriptions/product.asp?sku=ANSI+X9.63-2001
3. ISO/IEC 18033-2 Encryption algorithms - Part 2: Asymmetric ciphers
http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=37971&scopelist=PROGRAMME
4. PKCS #1 v2.1: RSA Cryptography Standard
ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf
5. FIPS 186-2 Digital Signature Standard (DSS)
http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf

4 Overview

4.1 Complexity vs. security

Public-Key cryptography allows for a flexible implementation than can support multiple levels of security:

4.1.1 Confidentiality

Each device server has a secret private key and a public key. The public key is used for wrapping key material sent to the device server. The private key is used by the device server to unwrap the keys sent to it. The entity wrapping the encryption key is assured that only the device server that owns the private portion of this public key can unwrap the encryption key.
At this level, the device server cannot authenticate the entity wrapping the encryption key or detect replay attacks. However, this mode is simple to implement and is an improvement over the current scheme (keys in plain-text). The weaknesses of this approach can be mitigated by relying on the lower protocols (e.g. ipsec for iSCSI of FC-SP for Fibre-Channel) or on physical security of the connection.
Note that integrity checking is included.

4.1.2 Confidentiality and Sender Authentication

In addition to its private/public key pair, each device server stores the public keys of the key wrapping entities from which it is authorized to receive encryption keys. While these public keys are not secret, the device server shall store them in a way that will prevent an attacker from modifying a public key or even injecting his own (such operations will grant the attacker the ability to send wrapped keys to the device server). In addition, there must be controls on how public keys are introduced. This may be done out of band in a vendor specific way (See also 4.3).
The device server's public key is used for the encryption operation of the wrapping, and the wrapping entity's private key is used for the signing operation of the wrapping. By verifying the signature, the device server is assured of the sender's identity.

At this level it is possible to add protection from replay attacks by requesting a random nonce from the device server prior to wrapping and including this value in the signing operation. At this time we believe that it is more effective to solve this problem by protecting the link between application client and device server by other means (such as IPsec for iSCSI).
Note that the same message format can be used in both modes. In the “confidentiality assurance only” mode, the nonce and signature fields may not be populated by wrapper, and are not check by the device server.

4.2 No protocol negotiation

Similar to the approach taken in 06-103r2, there will be no protocol negotiation. The client application will discover the wrapping format supported by requesting the Supported Key Formats page (see ssc3r03 8.5.2.5). The wrapping entity is required to support all formats.

4.3 Management of Public Keys

While the public keys are not secret, installing a public key in an entity authorizes that entity to receive from (in the case of device server) or send to (in the case of the key wrapping entity) the owner of that public key. Therefore this process should be controlled. Once installed, the public keys should be maintained in an authorization white list that is protected against unauthorized modification (but need not be confidential).

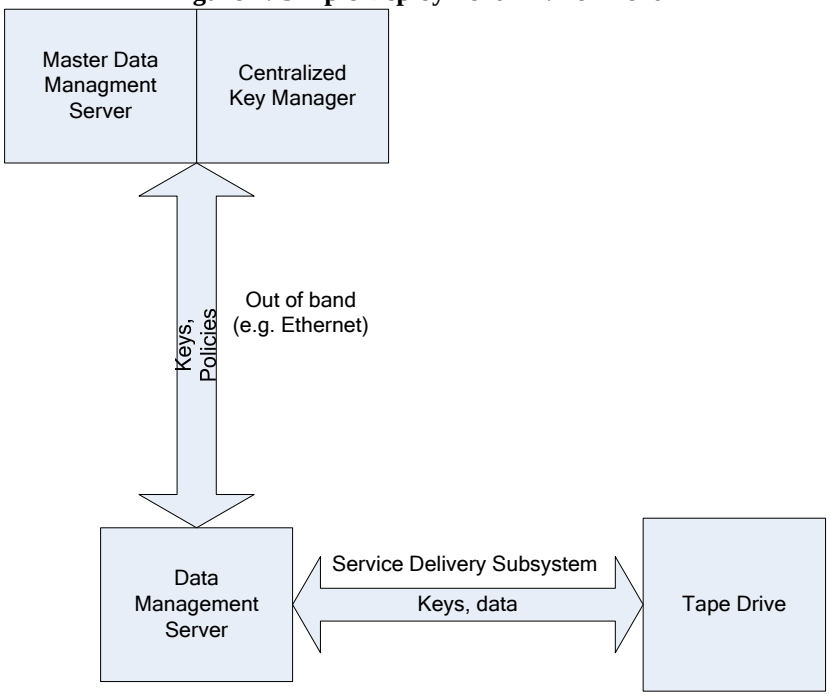
4.4 Threat Model Discussion

The goal of this section is to define the security environment, including the threats to the system, and to evaluate several proposals in light of this threat model.

4.4.1 Security Environment

The following diagram is meant to portray a typical deployment, in which a large number of tape devices are being accessed by a set of backup servers, each run by a master backup server. The addition of encryption in the tape drives introduces a Key Manager (which may or may not be part of the master backup server). Organizations will have a centralized key manager for the same reason that they use master backup servers, although there is no requirement that only one such is used, but rather that relatively few key managers are installed in an environment with a relatively large number of tape drivers and backup servers.

Figure 1: Simple Deployment Environment



Communication between the Key Manager/Backup Master and the Backup server is over the Ethernet network, while communication between the backup server and tape drive is over a network abstracted as "T10" (i.e. parallel-SCSI, FC, SCSI, iSCSI, etc.).

The following additional assumptions are in place:

1. The Ethernet network connection can easily be made sufficiently secure (e.g. via SSL with a 128 bit strong cipher suite is probably sufficient for most link based attacks)
2. The Key Manager will be able to reliably associate keys with Backup Servers
3. Only the backup server is able to associate keys with data attributes (e.g. tape pool)
4. The tape drive has a minimal cryptographic module, able to encrypt and decrypt data, and to perform simple key load/key agreement operations.
5. Attacks against key disclosure or substitution for keys already stored within the tape drive, or Key Manager are out of scope.
6. The Backup server, in most cases, will have only a simple SSL module implemented in the underlying OS.

As a result of the above, we assume threats are against either the backup server or the T10 network.

4.4.2 Threats

In the following table, the first column lists threats, while the first row lists function being attacked. Entries in the table represent mechanisms to mitigate the threat.

Table 1: Threats overview

Threat	*Against Backup Server*	*Service Delivery Subsystem*
Subverting key load protocol via CDB modification (e.g. turning off encrypted writes).	Harden the Backup Server OS of the backup manager	have a secure channel from the backup application to the tape drive.
Key Disclosure (passive)	Encrypt keys prior to reaching backup server. **or** Prevent information leakage by hardening the Backup Server OS to prevent caching or core dumps, etc.	Only send encrypted keys along T10 network, **or** have a secure channel from the backup application to the tape drive.
Injecting an attacker's key	Encrypt and sign keys at the key manager prior shipment to backup server. **or** Harden the Backup Server OS	Only send signed keys along the T10 network, **or** have a secure channel from the backup application to the tape drive.
Breaking association of keys to data via key replay	Harden the Backup Server OS	Rotate signing keys frequently, **or** have a secure channel from the backup application to the tape drive.

The most catastrophic threat is key disclosure (all other threats affect only a limited data window, and do not retroactively compromise encrypted tapes), and it can be mitigated by sending encrypted keys from the key manager to the Tape Drive, via the Backup Server. Other countermeasures are hardening the Backup Server to resist attempts to disable key encryption and using a policy at the Tape Drive that only accepts encrypted keys (i.e., will reject all unencrypted keys).

The next class of threats is either turning off data encryption or substitution of a legitimate data encrypting key with an attacker's key. The latter is more difficult to detect, but may require some reverse engineering skills; and both attacks require active tampering with the T10 network. To defend against them, a secure T10 channel may be used (e.g. IPsec in iSCSI). Optionally, keys may be signed by the key manager. In environments that which to encrypt all media, a policy on the device server could be used to ensure encryption can't be disabled.

The final class of threats assumes that an attacker has access to, or has compromised a valid key, and so wishes to read data by either substituting this (valid) key into a tape drive that processes someone else's data. Securing the T10 channel will protect against the substitution happening in the channel. The attack can also take place in the backup

server, but this is problematic, since attackers with this ability will already have access to the policies governing key to data association.

4.4.3 Proposed Solution

We propose encryption and signing of the keys at the key management server as this will protect against the most serious threat of key exposure. This approach provides protection of the keys without requiring a major overhaul of backup server technology (i.e. OS and application hardening…)
In this proposal we specify key wrapping using RSAES-OAEP with modulus length of 2048 and ECIES using ECC-521. The proposal includes the necessary fields to allow for future key wrapping schemes to be added in the future.
Protection against link based attacks (e.g. key replay and disabling of encryption) can be done today with existing link protection technology such as IPsec and FC-SP.

5 Proposed changes to SSC-3 (based on ssc3r03a)

5.1 Additions to 2.2 Approved references

ANSI 9.63-2001 Public Key Cryptography for the Financial Services Industry, Key Agreement and Key Transport Using Elliptic Curve Cryptography
ISO/IEC 18033-2 Encryption algorithms - Part 2: Asymmetric ciphers
PKCS #1 v2.1: RSA Cryptography Standard
FIPS 186-2 Digital Signature Standard (DSS)

5.2 Add 4.2.x Protection of Data Encryption Keys

4.2.x.1 Data Encryption Key Protection Overview

A device server that supports data encryption should protect data encryption keys from disclosure. Key disclosure may be mitigated by several countermeasures such as key wrapping and/or securing the channel used to transmit the key.

4.2.x.2 Key Wrapping Using Public Key Cryptography

A device server that supports public key cryptography for key wrapping, shall have a secret private key and a public key. The public key is used for wrapping key material sent to the device server. The private key is used by the device server to unwrap the keys sent to it. The entity wrapping the encryption key is assured that only the device server that owns the private portion of this public key can unwrap the encryption key.

A device server that supports public key cryptography for key wrapping may ensure the authenticity of the wrapped key by verifying the key wrapper's signature. The key wrapping entity's secret private key is used to sign the wrapped key. The key signing entity's public key is used by the device server to verify the signature.

A device server that supports signature verification shall store the key wrappers' public keys in an authorization white list. While these public keys are not secret, the device server shall maintain the authorization white list in a way that will prevent an attacker from modifying a public key or even injecting his own (such operations will grant the attacker the ability to send wrapped keys to the device server).

A device server that supports signature verification should be able to store a minimum of four public keys for signature verification to allow for two key wrapping entities and the ability to gracefully replace these keys.

The method of adding signature verification public keys to the authorization white list is beyond the scope of this standard.

5.3 8.5.2 SECURITY PROTOCOL IN command specifying Tape Data Encryption security protocol

5.3.1 Modify table 94

Code	Description	Reference
0000h	Tape Data Encryption In Support page	8.5.2.2
0001h	Tape Data Encryption Out Support page	8.5.2.3
0002h-000Fh	Reserved	
0010h	Data Encryption Capabilities page	8.5.2.4
0011h	Supported Key Formats page	8.5.2.5
0012h	Data Encryption Management Capabilities page	8.5.2.6
0013h-001Fh	Reserved	
0020h	Data Encryption Status page	8.5.2.7
0021h	Next Block Encryption Status page	8.5.2.8
0022h-00XXh	Reserved	
00XXh	Device Server Key Wrapping Public Key page	8.5.2.9
00XXh-FFFFh	Reserved	

5.3.2 Add 8.5.2.9

8.5.2.9 Device Server Key Wrapping Public Key page

This page is used by an application client to read the device server's key wrapping public key.

Table P – Device Server Key Wrapping Public Key page

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (0030h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4	(MSB)			PUBLIC KEY TYPE				
7								(LSB)
8	(MSB)			PUBLIC KEY FORMAT				
11								(LSB)
12	(MSB)			PUBLIC KEY LENGTH (n-9)				
13								(LSB)
14				PUBLIC KEY				
n								

The PUBLIC KEY TYPE field indicates the type of public key in the PUBLIC KEY field. Values for the PUBLIC KEY TYPE field are specified in table T.

Table T – PUBLIC KEY TYPE field values

Code	Description	Reference
00000000h	RSA 2048	8.5.2.9.1
00000001h-0000000Fh	Reserved	
00000010h	ECC 521	8.5.2.9.2
00000011h – FFFFBFFFh	Reserved	
FFFFC000h – FFFFFFFFh	Vendor Specific	

The PUBLIC KEY FORMAT, PUBLIC KEY LENGTH and PUBLIC KEY fields depend on the PUBLIC KEY TYPE field (see table T.)

8.5.2.9.1 RSA 2048 Public Keys

The PUBLIC KEY FORMAT field shall be set to 00000000h. All other values for the PUBLIC KEY FORMAT field are reserved. The PUBLIC KEY LENTGH field shall be set to 512. Bytes 14 through 269 shall contain the modulus n. Bytes 270 though 525 shall contain the public exponent e (See PKCS #1 V2.1).

8.5.2.9.1 ECC 521 Public Keys

The PUBLIC KEY FORMAT field shall be set to 00000000h. All other values for the PUBLIC KEY FORMAT field are reserved. The PUBLIC KEY LENTGH field shall be set to 133. Bytes 14 through 146 shall contain the ECC 521 public key as converted by the algorithm specified in ANSI X9.63 section 4.3.6 using the uncompressed form.

5.4 Changes in 8.5.3.2 Set Data Encryption page

5.4.1 Modify table 113:

Table 113 — KEY FORMAT field values

Code	Description	Reference
00h	The KEY field contains the key to be used to encrypt or decrypt data.	8.5.3.2.1
01h	The KEY field contains a vendor-specific key reference.	8.5.3.2.2
02h	The KEY field contains the key wrapped by the device server's public key	8.5.3.2.3
03h-BFh	Reserved	
C0h-FFh	Vendor specific	

5.4.2 Move key format descriptions into 8.5.3.2.1 and 8.5.3.2.2 and add 8.5.3.2.3 and 8.5.3.2.4

8.5.3.2.1 Plain-Text Key

If the KEY FORMAT field is set to 00h, the KEY field contains the key in an algorithm-specific format. Table 114 defines the format of the key in the KEY field if the KEY FORMAT field is set to 00h.

<< Note to Editor: move table 114 here >>

The KEY LENGTH field indicates the length of the KEY field in bytes.

8.5.3.2.2 Key Reference

If the KEY FORMAT field is set to 01h, the KEY field shall contain 8 bytes of T10 vendor identification (see SPC-4) followed immediately by a vendor-specific key reference identifying the key to be used to encrypt or decrypt data. If the KEY field contains a vendor-specific key reference that is unknown to the device server, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to VENDOR SPECIFIC KEY REFERENCE NOT FOUND.

8.5.3.2.3 Key Wrapped by Device Server's Public Key

If the KEY FORMAT field is set to 02h, the KEY field will consist of the encrypted key, a label and a signature. Table K defines the format of the KEY field.

Table K – KEY field contents with KEY FORMAT field set to 02h

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER SET						
1								(LSB)
2	(MSB)	LABEL LENGTH						
3								(LSB)
4	(MSB)	LABEL						
n								(LSB)
n+1	(MSB)	WRAPPED KEY LENGTH						
m								(LSB)
m+1	(MSB)	WRAPPED KEY						
l								(LSB)
l+1	(MSB)	SIGNATURE LENGTH						
k								(LSB)
k+1	(MSB)	SIGNATURE						
J								(LSB)

The PARAMETER SET field (see Table W) specifies the parameters used in wrapping operation.

Table W – PARAMETER SET field values

Code	Description	Reference
0000h	RSA 2048	8.5.3.2.3.1
0001h-000Fh	Reserved	
0010h	ECC 521	8.5.3.2.3.2
0011h – BFFFh	Reserved	
C000h – FFFFh	Vendor Specific	

The LABEL LENGTH field indicates the length of LABEL field in bytes.

The LABEL field contains public information associated with the data encryption key (e.g. Key identification). The LABEL field shall consist of a version byte and a format byte; both set to 0000h, followed by Wrapped key descriptors (see 8.5.4.6).

The WRAPPED KEY LENGTH field indicates the length of WRAPPED KEY field in bytes.

The WRAPPED KEY field contains the data encryption key encrypted by the parameters specified in the PARAMETER SET field.

The SIGNATURE LENGTH field indicates the length of SIGNATURE field in bytes.

The SIGNATURE field contains the wrapper's signature as specified by the PARAMETER SET field.

If the device server determines that the wrong public key was use to wrapped the data encryption key, the device server shall terminate the command with CHECK CONDITION status and set the sense key to DATA PROTECT and the additional sense code to INCORRECT DATA ENCRYPTION KEY.

If a device server that tries to verify the signature determines that it does not have the specified signature verification public key, the device server shall terminate the command with CHECK CONDITION status and set the sense key to DATA PROTECT and the additional sense code to UNKNOWN SIGNATURE VERIFICATION KEY.

Note to editor: UNKNOWN SIGNATURE VERIFICATION KEY is a new ASC.

If the device server encounters an error while unwrapping the data encryption key, the device server shall terminate the command with CHECK CONDITION status and set the sense key to DATA PROTECT and the additional sense code to UNABLE TO DECRYPT DATA.

If a device server encounters an error while verifying the signature over the wrapped data encryption key, the device server shall terminate the command with CHECK CONDITION status and set the sense key to DATA PROTECT and the additional sense code to CRYPTOGRAPHIC INTEGRITY VALIDATION FAILED.

8.5.3.2.3.1 Key Wrapping with RSA 2048

If the PARAMETER SET field is set to 0000h, the WRAPPED KEY field shall contain the data encryption key wrapped using RSAES-OAEP as specified by PKCS #1 v2.1. The RSA modulus length shall be 2048 bits. The hash function used shall be SHA-256. The mask generation function (MGF) used shall be MGF1 with SHA-256 as the hash function.

The WRAPPED KEY shall be the data encryption key encrypted with the device server's public key as specified by PKCS #1 v2.1 in 7.1.1 RSAES-OAEP-ENCRYPT ((n, e), M, L), where (n, e) is the public key, M is the raw key and L is the LABEL field as described above.

If a signature is included, the SIGNATURE field shall be the RSASSA-PSS signature over the WRAPPED KEY field as specified by PKCS #1 v2.1 in 8.1.1 using the wrapping entity's private key. The RSAES-OAEP-ENCRYPT operation adds an integrity check to both L (the label) and M (the key). The signature is for providing proof of the wrapper's identification. The device server may check the signature to verify that the key was wrapped and signed by a valid key management server.

8.5.3.2.3.2 Key Wrapping with ECC-521

If the PARAMETER SET field is set to 0010h, WRAPPED KEY field shall contain the data encryption key wrapped using ECIES-HC as specified in ISO/IEC 18033-2. The following parameters shall be used for ECIES-HC:

- ECIES-KEM
  - The elliptic curve used shall be Curve P-521 as specified in appendix 6 of FIPS 186-2 Change 1.
  - KDF shall be the key derivation function defined by Approved Alternative 1 as specified in NIST SP800-56A. The KDF parameters are defined below:
    - H shall be SHA-512
    - AlgorithmID shall be set to 00001h
    - PartyUInfo shall be the device server identification (see 8.5.4.7)
    - PartyVInfo shall be the wrapper identification (see 8.5.4.8)
    - SuppPubInfo and SuppPrivInfo shall not be included.
  - CofactorMode = 0
  - OldCofactorMode = 0
  - CheckMode = 1
  - SingleHashMode = 0
  - KeyLen = 96
  - *fmt* shall be uncompressed
- DEM1
  - SC is SC1 using AES-256 as the block cipher
  - MA is HMAC with SHA-512

If a signature is included, the SIGNATURE field shall be the ECDSA signature over the WRAPPED KEY field as specified by FIPS 186-2 using the wrapping entity's private key. The device server may check the signature to verify that the key was wrapped and signed by a valid key management server.

Note: The deviation of the definition of the KDF from the ISO standard is to allow conforming implementation to be FIPS 140-2 compliant.

5.5 New sections inserted after 8.5.4.5

8.5.4.6 Wrapped Key Descriptors

Each Wrapped key descriptor shall be of the format as specified in table A.

Table A — Wrapped Key Descriptor Format

Bit / Byte	7	6	5	4	3	2	1	0
0	Wrapped Key Descriptor Type							
1	Reserved							
2	(MSB)	Wrapped Key Descriptor Length (n-3)						
3								(LSB)
4	(MSB)	Wrapped Key Descriptor						
N								(LSB)

If more than one wrapped key descriptor is specified, they shall be in increasing numeric order of the value in the Wrapped Key Descriptor Type field.

The Wrapped key descriptor type field contains a value from table B that defines the contents of the Wrapped key descriptor.

Table B – Wrapped key Descriptor field values

Code	Description	Reference	Required
00h	Device Server Identification	8.5.4.7	Yes
01h	Wrapper identification	8.5.4.8	Yes
02h	Key Label	8.5.4.9	
03h	Key identification	8.5.4.10	Yes
04h	Key Length	8.5.4.11	Yes
05-BFh	Reserved		
C0-FFh	Vendor Specific		

8.5.4.7 Device Server Identification descriptor

The Wrapped Key Descriptor field shall contain the logical unit name (see SAM-4) for the device server. This value may be used by the device server to validate that the data encryption key was wrapped with the correct public key. It may also be used for this by application client in the case were the key was wrapped be a third system (e.g. a key management server).

8.5.4.8 Wrapped Key Wrapper identification descriptor

The Wrapped Key Descriptor field shall contain identification data that uniquely identify the key wrapping entity. This value may be used by the device server to locate the wrapper’s public key for signature verification.

8.5.4.9 Wrapped Key Label descriptor

The Wrapped Key Descriptor field may be used to identify the key. This information is not required to be unique (e.g. a label entered by the system operator).

8.5.4.10 Wrapped Key identification descriptor

The Wrapped Key Identification descriptor field shall contain the key identifier.

8.5.4.11 Wrapped Key Length descriptor

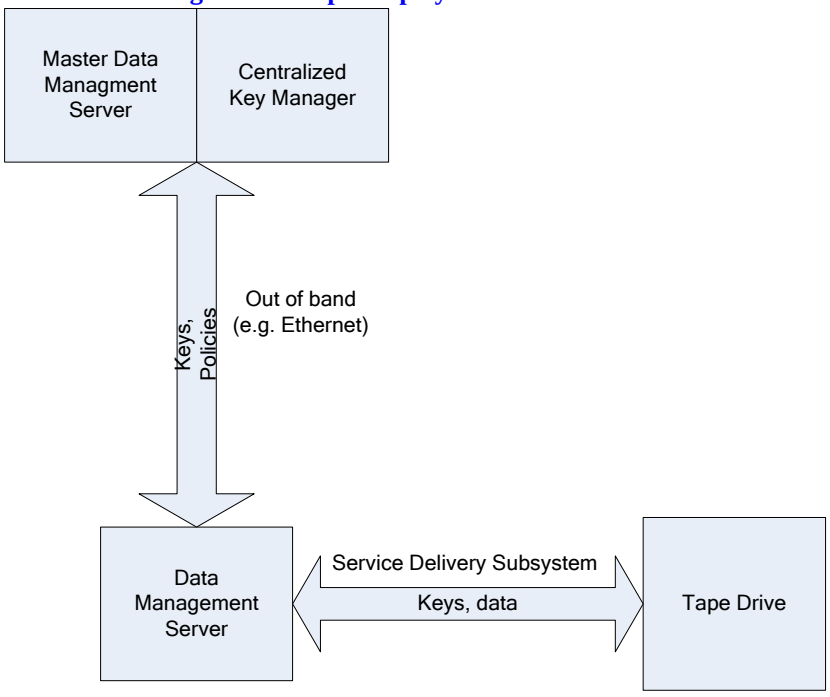
The Wrapped Key Length descriptor field shall contain the length of the wrapped key in bytes.

5.6 Add in informative annex:

***** Security Environment**

The following diagram is a simplified depiction of a typical deployment, in which a large number of device servers are being accessed by a set of data management servers, each set controlled by a master data management server. The addition of encryption in the device servers introduces a Key Manager (which may or may not be part of the master data management server). Centralized key managers are used for the same reason that they use master data management servers, so that relatively few key managers are installed in an environment with a relatively large number of device servers and data management servers.

Figure X: Simple Deployment Environment



Communication between the Key Manager/Master Server and the data management server is beyond the scope of this standard, while communication between the data management server and device server is over a SCSI service delivery subsystem.

The following additional assumptions are in place:

1. The communications path between the key manager and the data management server can easily be made sufficiently secure (e.g. via SSL with a 128 bit strong cipher suite is probably sufficient for most link based attacks)
2. The Key Manager is able to reliably associate keys with Data Management Servers
3. Only the data management server is able to associate keys with data attributes
4. The device server has a minimal cryptographic module, able to encrypt and decrypt data, and to perform simple key load/key agreement operations.
5. Attacks against key disclosure or substitution for keys already stored within the device server, or Key Manager are out of scope.
6. The data management server, in most cases, has only a simple SSL module implemented in the underlying OS.

As a result of the above, this model assumes threats are against either the data management server or the service delivery subsystem.

*** Threats

Table 2 contains a list of security threats and methods to mitigate these threats. The first column lists threats, while the first row lists function being attacked. Entries in the table represent mechanisms to mitigate the threat.

Table 2: Threats overview

Threat	*Against Data Management Server*	*Against Service Delivery Subsystem*
Subverting key load protocol via CDB modification (e.g. turning off data encryption).	Harden the OS of the data management server	have a secure channel from the data management server to the device server.
Key Disclosure (passive)	Encrypt keys prior to reaching the data management server. **or** Prevent information leakage by hardening the OS of the data management server to prevent caching or core dumps, etc.	Only send encrypted keys along service delivery subsystem network, **or** Have a secure channel from the data management server to the device server.
Injecting an attacker's key	Encrypt and sign keys at the key manager prior shipment to data	Only send signed keys along the T10 network, **or**

	management server. **or** Harden the OS of the data management server	Have a secure channel from the data management server to the device server.
Breaking association of keys to data via key replay	Harden the OS of the data management server	Rotate signing keys frequently, **or** Have a secure channel from the data management server to the device server.

A particularly catastrophic threat is key disclosure (all other threats affect only a limited data window, and do not retroactively compromise encrypted media), and it can be mitigated by sending encrypted keys from the key manager to the device server, via the data management server. Other countermeasures are hardening the data management server to resist attempts to disable key encryption and using a policy at the device server that only accepts encrypted keys (i.e., will reject all unencrypted keys).

The next class of threats is either turning off data encryption or substitution of a legitimate data encrypting key with an attacker's key. The latter is more difficult to detect, but may require some reverse engineering skills; and both attacks require active tampering with the delivery subsystem. To defend against them, a secure channel may be used (e.g. IPsec in iSCSI). Optionally, keys may be signed by the key manager. In environments that wish to encrypt all media, a policy on the device server could be used to ensure encryption can't be disabled.

The final class of threats assumes that an attacker has access to, or has compromised a valid key, and so wishes to read data by either substituting this (valid) key into a tape drive that processes someone else's data. Securing the service delivery subsystem protects against the substitution happening in the channel. It is also possible for this attack to take place in the data management server, but this is problematic, since attackers with this ability already have access to the policies governing key to data association.