To: IEEE P1619.3 Task Group
From: P1619.3 [subgroup or ad hoc committee], Members:
[Subhash Sankuratripati, NetApp]
[Ravi Kavuri, NetApp]
[Gaurav Agarwal, NetApp]
[Scott Kipp, Brocade]
[Landon Noll, Cisco]
[Kevin Marks, Dell]
[Jon Hass, Dell]
[Larry Hofer, Emulex]
[Glen Jaquette, IBM]
[Walt Hubis, LSI]
[Jon Holdman, Sun]
[Luther Martin, Voltage Security]
[Matthew Ball, MV Ball Consulting]
[Robert A. (Bob) Lockhart, nCipher]

Date: May 4, 2008
Purpose: Proposed changes against P1619.3/D3 to incorporate [Add Sections 4, 5 & 6 into the draft._]
[NOTE: Draft number proposed against should replace red x]

Introduction

The P1619.3 [subgroup or ad hoc committee] has been working on a proposal to create [Fill in an overview of what the committee is proposing].

[Please delete anything between brackets (including the brackets) and replace with the appropriate proposals]

[Rules and Guidance

a) *Diagrams can be submitted in color or grayscale. You may be asked to convert it if time is not on the editor's side at the moment.*

b) *All* [black bracketed] *text should be replaced with the appropriate information.*

i) *Note please delete this bullet completely. The format of the text is there for example purposes.*

c) *All text in dark red italics should be cut out of a document prior to submission (includes this entire section with numbering).*

d) *If you have verbiage or information that belongs in a section that Bob L. did not include you as creating, ignore Bob L.(this once only) and create away!*

e) *All dates that must be updated are automatically updated as you open & save the document. You should replace them with fixed dates for proposal and tracking purposes.*

f) *If you are not running a PC with Windows using Word, do not enable the macros. You should also avoid deleting them. Any documents that have to be cut and pasted back into a good macro document will cost you $25 to be put towards the WTSS subgroup (see P1619.3 meeting minutes dated September 17 2007).*

g) *Bob Lockhart will be participating off and on in all subgroups to monitor progress and assist with the documents as needed.]*

Changes to P1619.3/D3

[Change the red x to the appropriate draft number]

[Note any sections that are to be completely removed from the existing document.]
[Note sections that contain changes if the section is not to be fully deleted.]

1. Normative References

[Include any normative references in this section]

2. Definitions, acronyms, and abbreviations

For the purposes of this proposal, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standards, Seventh Edition,* should be referenced for terms not defined in this clause.

[Ensure that definitions, acronyms and abbreviations do not already exist in the above reference]

2.1 Definitions

***[3.1.1.term1:** select the 'definitions' and select 'format terms and definitions' in the IEEEStdsTemplate tool bar. If you do not have macro enabled please enter the number manually.]*

2.2 Acronyms and abbreviations

[LAH List Acronyms (and/or abbreviations) Here]

[I have to manually edit the numbers here so just use standard body text formatting.]

3. General Overview

[Place any overview information as it pertains to the proposal in this section. Use section numbers appropriately. The editor reserves the right to move information from any section to a more appropriate section based on workgroup feedback]

[Architecture and Name Space belong in this section]
[We may need to move some or all of Name Space to section 5 or give it a separate section]

4. Key Management Objects

4.1 Key

Scope: Client & Server.

The Key object consists of the key blob (potentially wrapped) along its meta-data.

4.1.1 Attributes

A key object contains the following attributes:

- SO_GUID
- STATE (Type:unsigned int – as defined in the architectural specification)
- T_EXPIRED (Type: UTC - time beyond which the key should not be used to encrypt new data)
- T_DISABLED (Type: UTC - time beyond which the key should be used)
- T_CACHED (Type: 64-bits – seconds that the key may be cached for)
- CIPHER_TYPE (Type:String as defined in KeyBlob section)
- KEY_BLOB (Object as defined in the next section)
- *WRAPPINGKEY_SO_GUID (IF KEY IS A MANAGED KEY)*
- *WRAPPING_KEY_ID (NON-MANAGED KEY)*
- *VENDOR_SPECIFIC_EXTENSIONS*
- *APPLICATION_EXTENSIONS*

4.1.2 States

As defined in the key state diagram (Editorial: as defined by the architecture sub-committee)

4.1.3 Operations

- Create
- Get
- Store

4.2 Key Blob

4.2.1 Attributes

— Version (Type:int and defined as 1 for this version of the standard).

— Length (Type:int)

— Data (Type: Character Array)

4.2.2 Blob Structure for Mandatory Cipher Types

The standard defines the blob format for the following mandatory cipher types:

Editorial: Unable to create tables using the IEEE add-in.

Cipher	Data
XTS-AES-128	Base64_encode (32-byte key)
XTS-AES-256	Base64_encode (64-byte key)
CCM-128-AES-256	Base64_encode (32-byte key)
GCM-128-AES-256	Base64_encode (32-byte key)
CBC-AES-HMAC-SHA-1	Base64_encode (52-byte key)
CBC-AES-HMAC-SHA-256	Base64_encode (64-byte key)
CBC-AES-HMAC-SHA-512	Base64_encode (96-byte key)
XTS-AES-HMAC-SHA-512	Base64_encode (128-byte key)
AES-128	Base64_encode (16-byte key)
AES-192	Base64_encode (24-byte key)
AES-256	Base64_encode (32-byte key)
3-DES	Base64_encode (24-byte key)
BitLocker	48-byte recovery key

NOTE: Cipher Type ID’s are reserved from 0000 0000 – 0100 0000. Vendor specific ID’s start from 1000 0000 – 1111 1111

4.2.3 Key Wrapping

Keys being transmitted via KMCS operations may (should is not the right word here) be wrapped. Wrapping can be used to add an additional layer of security (apart from the transport level security) or to ensure that the KM client does not have access to the key material. This is useful in the case where the CU (cryptographic unit) and the KM have established a trust mechanism (the means of which is outside the purview of this standard).

Keys when wrapped SHALL be signed along with certain meta-data attributes to ensure the CU can be assured of the integrity of the package.

Signature: The signature SHALL consist of the encrypted keyblob.data along with a hash of the following attributes and in the sequence laid out:

- SO_GUID.SO_HANDLE (just the handle and not the entire SO_GUID. This is to ensure that a CU need not be aware of the entire SO_GUID construction mechanisms).
- T_EXPIRED
- T_DISABLED
- T_CACHED
- CIPHER_TYPE

The KM client can compute a hash of these constituents and send it to the CU or it can send the list of attributes along to the CU which can then compute the hash value. The former shall be used in scenarios where the CU is agnostic of these attributes (a dumb CU) and the later in cases where the CU understands these attributes.

Signature = Signature-Algorithm (Encrypted Key Material || Hash (SO_HANDLE || T_EXPIRED || T_DISABLED || T_CACHED || CIPHER_TYPE))

NOTE: The hash function SHALL be either SHA-256 or SHA-512.

For e.g. when the case where the Signature-Algorithm is HMAC-SHA-512 and the hash function is SHA-512, the signature would be computed as follows:

Hash-Value = SHA-512(SO_HANDLE || T_EXPIRED || T_DISABLED || T_CACHED || CIPHER_TYPE)

Signature = HMAC-SHA-512[SS](Encrypted Key Material || Hash-Value) where SS is a shared secret that has been established with the KM.

A KM SHALL support the following wrapping algorithms:

1. AES-KeyWrap
2. CBC-AES-HMAC-SHA(256 or 512)

A KM SHOULD support the following wrapping algorithms:

1. RSA-OAEP / DSA.
2. RSA-OAEP / RSA-PSS.
3. ECIES / ECDSA.

4.3 Client

Scope: Client & Server.

The client object consists of its credentials and capabilities.

4.3.1 Attributes

- CREDENTIAL_TYPE (Session, Username/Password, Symmetric / Asymmetric key, SSO, CHAP)
- List of {CREDENTIAL_LENGTH,CREDENTIAL_VALUE} tuples.

4.3.2 States

- Active
- Disabled/Locked
- Authenticated

4.3.3 Operations

- Create
- Delete
- Authenticate
- Disable

NOTE: All of these operations with the exception of authenticate are performed by way of KM Console operations and are outside the scope of the standard.

4.4 Capability

4.4.1 Attributes

- Name (Type: String)

4.4.2 States

None.

4.4.3 Operations

No direct operations. The capability object is sent as part of the capability negotiation operation.

4.5 Key Manager

Scope: Server only.

NOTE: In this version of the specification, a key manager is the same as a client as there no operations that are KM ⇔ KM specific.

4.6 Data Sets

Scope: Client & Server.

Data sets represent manageable units of encrypted data. Data sets are expressed as selection rules that can be applied to data set attributes such as file path, tape volume id, server IP, or a range of disk blocks. There should be flexibility in defining what a data set is, depending on the position of the encryption agent "in the stack" of the storage infrastructure.

Once the data sets are identified, keys may be associated to data sets via a key assignment policy.

4.6.1 Attributes

- NAME
- VALUE
- SIZEOF_VALUE

4.7 Client Groups

Scope: Server only.

Clients may be grouped together for ease of management. This grouping may be static – i.e. clients are explicitly added into a group or dynamic i.e. based on a regular expression match on client attributes.

4.7.1 Attributes

- TYPE (STATIC or DYNAMIC)
- List of CLIENT_SO_GUIDs (only in case of static binding)
- List of {PATTERN, ATTRIBUTE} tuple.

4.8 Key Groups

Scope: Server only.

Keys may be grouped together for ease of management. This grouping may be static – i.e. explicitly added into a group or dynamic i.e. based on a regular expression match on dataset attributes.

4.8.1 Attributes

- TYPE (STATIC or DYNAMIC)
- List of CLIENT_SO_GUIDs (only in case of static binding)
- List of {PATTERN, DATASET} tuple.

5. Key Management Policies

A policy is a deliberate plan of action to guide decisions and achieve rational outcome(s). (Source: Wikipedia). In the same vein, Key Management Policies are used to guide assignment, retention, wrapping, replication & access control decisions on keys.

The scope of all policy objects are ‘Server only’.

5.1 Key Assignment Policy

Key Assignment Policies contain logic that is able to determine which data set should be encrypted with which key using which algorithm (e.g. encrypt and sign all emails sent outside of the company. Sign all tapes with a unique key per tape, etc.)

A key assignment policy determines

— the type of unencrypted data that is determined to be encrypted with a specific key.

— how often to generate new keys

Therefore, the key assignment policy encapsulates both key generation and key scope policies. This is done to fit regular usage patterns. For example, when a tape is loaded into a drive, the drive will request a key by data, and will receive both a key that may be used on that drive, as well as a policy notifying the drive whether all tapes should be encrypted with this key, or only the current tape.

The key assignment policy is determined by the set of supported data set attributes, and is encoded as a set of name, value pairs.

The policy is interpreted to mean that the key may be used whenever all the named parameters have values equal to the values of the data presented to the client. So, for example, in the following encryption policy:

container attribute name = ”storage_server_name” value=”Ireland.com”

data attribute name = ”financial”

The provided key may be used to encrypt all of the data tagged as “financial” on the storage server named “Ireland.com” with the key listed in the KeyExchangeStructure.

Note that there is a difference between a data set binding and an assignment policy, and the KMS must track both. An organization may set a policy to encrypt a pool with a specific key, but due to key rotation policies, not all tapes within the pool will have been encrypted with that key. Therefore, assignment policies specify the current desired behavior, whereas the KMS will store all data set bindings that are reported to it, for future audit and key query commands. State logic within the KMS may allow for the automatic creation of key assignment policies based on the “most recent” data set binding.

5.2 Retention Policy

5.2.1 Overview

The retention policy dictates the duration for which protected data, hence the key with which it is encrypted, is accessible to a given client. It also dictates when new data should no longer be encrypted with a given key.

This policy should be superseded by the key life cycle.

5.2.2 Attributes

- T_EXPIRATION
- T_DISABLE

5.2.3 States

- Created
- Assigned
- Enforcing
- Disabled

5.2.4 Operations

- Add
- Associate
- Disassociate
- Delete

5.3 Wrapping Policy

The wrapping policy indicates whether a key should be wrapped prior to being dispatched to a client. This policy may be applied either at the key level or at a client level. If a key has a wrapping policy, then it SHOULD override the client's wrapping policy.

5.3.1 Attributes

- WRAPPING_TYPE (asymmetric / symmetric / signature)
- WRAPPING_MODE (as listed in the key blob section)

5.3.2 States

- Created
- Assigned
- Enforcing
- Disabled

5.3.3 Operations

- Add
- Associate
- Disassociate
- Delete

5.4 Audit Policy

Audit policies state the auditory requirements that need to be enforced on keys and clients.

TODO: Bob Lockhart to provide new content.

5.4.1 Attributes

- Operation type
- Event type (to trigger, Log, SNMP trap, etc.) Mandatory: Log
- Event Dispatch Destination (Local, SNMP, syslog) Mandatory: Local, syslog.

NOTE: The only mandatory type that should be supported is ‘log’ and the mandatory dispatch destinations are local and syslog.

5.4.2 States

- Created
- Assigned
- Enforcing
- Disabled

5.4.3 Operations

- Add
- Associate
- Disassociate
- Delete

5.5 Replication Policy

The key replication policy describes the set of key management servers the keys should be replicated to. The synchronization protocol is outside the scope of this version of the standard since KMSS operations are out of scope.

It is assumed that a KM server would act as a ‘trusted client’ and utilize KMCS operations to synchronize keys.

5.5.1 Attributes

— List of KM servers

5.5.2 States

— Created
— Assigned
— Enforcing
— Disabled

5.5.3 Operations

— Add
— Associate
— Disassociate
— Delete

5.6 Access/Distribution Policy

Key access policies encode which clients and key management servers may access which keys. This is implicitly controlled by the clients, as the clients set the data set bindings of keys, but it is enforced by the KMS, which lookup tables storing keys to data assignments, and clients to data permissions.

This policy is applied at a key/key group level.

5.6.1 Attributes

- SO_GUID
- Client list
- KM server list.

5.6.2 States

- Created
- Assigned
- Enforcing
- Disabled

5.6.3 Operations

- Add
- Associate
- Disassociate
- Delete

5.7 Caching Policy

The key caching policy dictates whether a key shall be cached by a KM client and if so, the duration for which it can.

Attributes

— T_CACHE_INTERVAL

5.7.1 States

— Created
— Assigned
— Enforcing
— Disabled

5.7.2 Operations

— Add
— Associate
— Disassociate
— Delete

6. Key Management Operations

6.1 Authenticate

Scope: KMCS

6.1.1 Overview

A client needs to authenticate with the KM server to perform any sensitive operations. Authentication is accomplished either at the transport level (SSL/TLS) or at the object/messaging level. Every request shall contain the “credential” object so that the KM server can validate the client.

6.1.2 Input / Output / Error

- (I): Client
- (O): Credentials (If the request type is login and not validation)
- (E): E_INVALID_CREDENTIALS
- (E): E_UNSUPPORTED_AUTHENTICATION_MODE

6.2 Capability Negotiation

Scope: KMCS

6.2.1 Overview

The client sends its capabilities to the server and the server returns back a list of capabilities it supports. If none of the capabilities are supported, then it returns back an empty list.

6.2.2 Input / Output / Error

- (I): Client
- (I): List of Capability Objects
- (O): List of Capability Objects that are supported by the KM server

6.3 Get Server Capabilities

- (I): Client
- (O): List of Capability Objects.

6.4 Create/Generate Key

Scope: KMCS, KM Console

6.4.1 Overview

A client upon authentication invokes the generate key operation to generate a new key by passing in the DataSet context in which this key would be used so that the KM can apply the appropriate policies.

6.4.2 Input / Output / Error

- (I): Client
- (I): List of Dataset objects.
- (O): Key
- (E): …

6.5 Store Key

Scope: KMCS, KM Console

6.5.1 Overview

Keys that are generated at the client can be stored in the KM server by invoking its store functionality.

6.5.2 Input / Output / Error

(I): Client

(I): List of Dataset objects.

(O): Key_SO_GUID

(E)…

6.6 Get Key

6.6.1 Overview

Clients invoke the get key operation to fetch keys from the KM server. They may invoke the query based on either a Key ID or based on the DataSet attributes. When querying based on dataset attributes, the KM returns a key based on the application template and the policies that govern the key and the client.

6.6.2 Input / Output / Error

(I): Client

(I): List of Dataset objects.

(O): Key

(E): …

6.7 Push Audit Message

6.7.1 Overview

This operation is intended to be used by ‘super’ clients that maintain local caches of keys and ship them out to cryptographic units on demand. This will ensure that the KM can be a central audit repository for any/all accesses to keys.

6.7.2 Input / Output / Error

(I): Client

(I): Key_SO_GUID

(I): Message (Optional)

(O): Boolean – SUCCESS/FAILURE.

(E): …

6.8 Get Random Bytes

6.8.1 Input / Output / Error

- (I): Client
- (I): numbers of bytes desired
- (O) Base64 Encoded bytes

7. Key Management Transport

[Supported transport protocols go here]

8. Key Management Messaging

[This is an additional section that I am proposing we add to help complete the standard. It does not currently exist in Draft 1.]
[This section would contain normative information for the XML and/or TLV formats we decide to support]

Annex A

(informative)

Bibliography

[List all bibliographic material here]

Annex B (informative)

Example Use Cases

[Objects & operations or use cases should add the appropriate use cases here]

Annex C (informative)

XML and TLV Schema Definitions

C.1 XML Schema

[Additional messaging group information for selected XML syntax goes here]

C.2 TLV Schema

[Additional messaging group information for selected TLV schema goes here]