

1 To: IEEE P1619.3 Task Group
2 From: P1619.3 [subgroup or ad hoc committee], Members:
3 [Subhash Sankuratripati, NetApp]
4 [Ravi Kavuri, NetApp]
5 [Gaurav Agarwal, NetApp]
6 [Scott Kipp, Brocade]
7 [Landon Noll, Cisco]
8 [Kevin Marks, Dell]
9 [Jon Hass, Dell]
10 [Larry Hofer, Emulex]
11 [Glen Jaquette, IBM]
12 [Walt Hubis, LSI]
13 [Matthew Ball, MV Ball Consulting]
14 [Robert A. (Bob) Lockhart, nCipher]
15 [Jon Holdman, Sun]
16 [Luther Martin, Voltage Security]
17 [Michael Marcil, Voremetric]

18 Date: June 11, 2008

19 Purpose: Proposed changes against P1619.3/D3 to incorporate [Add Sections 4, 5 & 6 into the draft.]

20 **[NOTE: Draft number proposed against should replace red x]**

21 **Introduction**

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

24
25 *[Please delete anything between brackets (including the brackets) and replace with the appropriate
26 proposals]*

27
28 *[Rules and Guidance*

- 29 a) *Diagrams can be submitted in color or grayscale. You may be asked to convert it if time is not
30 on the editor's side at the moment.*
- 31 b) *All [black bracketed] text should be replaced with the appropriate information.*
- 32 i) *Note please delete this bullet completely. The format of the text is there for example
33 purposes.*
- 34 c) *All text in dark red italics should be cut out of a document prior to submission (includes this
35 entire section with numbering).*

- 1 d) *If you have verbiage or information that belongs in a section that Bob L. did not include you as*
2 *creating, ignore Bob L.(this once only) and create away!*
- 3 e) *All dates that must be updated are automatically updated as you open & save the document.*
4 *You should replace them with fixed dates for proposal and tracking purposes.*
- 5 f) *If you are not running a PC with Windows using Word, do not enable the macros. You should*
6 *also avoid deleting them. Any documents that have to be cut and pasted back into a good macro*
7 *document will cost you \$25 to be put towards the WTSS subgroup (see P1619.3 meeting minutes*
8 *dated September 17 2007).*
- 9 g) *Bob Lockhart will be participating off and on in all subgroups to monitor progress and assist*
10 *with the documents as needed.]*

11 **Changes to P1619.3/D3**

12 *[Change the red x to the appropriate draft number]*

13 *[Note any sections that are to be completely removed from the existing document.]*

14 *[Note sections that contain changes if the section is not to be fully deleted.]*

15 **1. Normative References**

16 *[Include any normative references in this section]*

17 **2. Definitions, acronyms, and abbreviations**

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

20

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

22 **2.1 Definitions**

23 *[3.1.1.term1: select the 'definitions' and select 'format terms and definitions' in the IEEEStdTemplate tool*
24 *bar. If you do not have macro enabled please enter the number manually.]*

25 **2.2 Acronyms and abbreviations**

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

27

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

29 **3. General Overview**

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

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

3 **4. Key Management Objects**

4 This section describes KM objects as they are transmitted across the wire to a KM client.

5 **4.1 Key**

6 **Scope:** Client & Server.

7 The Key object consists of the key blob (potentially wrapped) along its meta-data. Any object attributes
8 listed in '*italicized font*' are optional.

9 **4.1.1 Attributes**

10 A key object contains the following attributes:

- 11 — SO_GUID
- 12 — STATE (Type:String) EDITORIAL: Reference back to the relevant section.
- 13 — T_EXPIRED (Type: UTC - time beyond which the key should not be used to encrypt new data)
- 14 — T_DISABLED (Type: UTC - time beyond which the key should be used)
- 15 — T_CACHED (Type: 64-bits – seconds that the key may be cached for)
- 16 — CIPHER_TYPE (Type:String as defined in KeyBlob section)
- 17 — KEY_BLOB (Object as defined in the next section)
- 18 — *CU_ID (Type:String) – How this ID is derived is outside the scope of the standard.*
- 19 — *WRAPPINGKEY_SO_GUID (IF KEY IS A MANAGED KEY)*
- 20 — *WRAPPING_KEY_ID (NON-MANAGED KEY)*
- 21 — *VENDOR_SPECIFIC_EXTENSIONS*
- 22 — *APPLICATION_EXTENSIONS*

23 **4.1.2 States**

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

25 **4.1.3 Operations**

- 26 — Create
- 27 — Get
- 28 — Store
- 29

1 **4.2 Key Blob**

2 **4.2.1 Attributes**

- 3 — Version (Type:int and defined as 1 for this version of the standard).
- 4 — WRAPPING_TYPE (Type:String) – and can take any of the values as listed in the key wrapping
- 5 section.
- 6 — Length (Type:int)
- 7 — Data (Type: Character Array)

8 **4.2.2 Blob Structure for Cipher Types**

9 The standard defines the blob format for the cipher types that are defined in the table below.

10 NOTE: Support for these ciphers is not mandatory for clients.

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

Cipher	Data	Definition
XTS-AES-128	Base64_encode (32-byte key)	IEEE P1619
XTS-AES-256	Base64_encode (64-byte key)	IEEE P1619
CCM-128-AES-256	Base64_encode (32-byte key)	IEEE P1619.1
GCM-128-AES-256	Base64_encode (32-byte key)	IEEE P1619.1
CBC-AES-HMAC-SHA-1	Base64_encode (52-byte key)	IEEE P1619.1
CBC-AES-HMAC-SHA-256	Base64_encode (64-byte key)	IEEE P1619.1
CBC-AES-HMAC-SHA-512	Base64_encode (96-byte key)	IEEE P1619.1
XTS-AES-HMAC-SHA-512	Base64_encode (128-byte key)	IEEE P1619.1
AES-128	Base64_encode (16-byte key)	FIPS 198
AES-192	Base64_encode (24-byte key)	FIPS 198
AES-256	Base64_encode (32-byte key)	FIPS 198
3-DES	Base64_encode (24-byte key)	FIPS 46-3
BitLocker	Base64_encode (48-byte recovery key)	Search for BitLocker on msdn.microsoft.com
Custom	Custom	Vendor specific

12

1 **4.2.3 Key Wrapping**

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

7 The WRAPPING_TYPE attribute of the KEY BLOG object may take any of the following values:

- 8 — NONE
- 9 — AES_KEYWRAP
- 10 — CBC-AES-128-HMAC-SHA-256
- 11 — CBC-AES-256-HMAC-SHA-512
- 12 — HMAC-SHA-256
- 13 — HMAC-SHA-512
- 14 — OTHERS.

15 NOTE: All wrapping types other than NONE are optional since wrapping itself is optional.

16

17 **EDITORIAL: Once we have signoff on this aspect, I will provide the rest of the details.**

18

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

- 21 ~~——SO_GUID.SO_HANDLE (just the handle and not the entire SO_GUID. This is to ensure that a CU~~
22 ~~need not be aware of the entire SO_GUID construction mechanisms).~~
- 23 ~~——T_EXPIRED~~
- 24 ~~——T_DISABLED~~
- 25 ~~——T_CACHED~~
- 26 ~~——CIPHER_TYPE~~

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

31 ~~Signature = Signature Algorithm (Encrypted Key Material || SO_HANDLE || T_EXPIRED ||~~
32 ~~T_DISABLED || T_CACHED || CIPHER_TYPE)~~

33 ~~Signature = HMAC-SHA-512[SS](Encrypted Key Material || SO_HANDLE || T_EXPIRED ||~~
34 ~~T_DISABLED || T_CACHED || CIPHER_TYPE) where SS is a shared secret that has been established~~
35 ~~with the KM.~~

36 **4.3 Client**

1 **Scope:** Client & Server.

2 The client object consists of its credentials and capabilities.

3 **4.3.1 Attributes**

4 — CREDENTIAL_TYPE (Session, Username/Password, Symmetric / Asymmetric key, SSO, CHAP)

5 — List of {CREDENTIAL_LENGTH,CREDENTIAL_VALUE} tuples.

6 **4.3.2 States**

7 — Active

8 — Disabled/Locked

9 — Authenticated

10 **4.3.3 Operations**

11 — Create

12 — Delete

13 — Authenticate

14 — Disable

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

17 **4.4 Capability**

18 **4.4.1 Attributes**

19 — Name (Type: String)

20 **4.4.2 States**

21 None.

22 **4.4.3 Operations**

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

24

1 **4.5 Key Manager**

2 **Scope:** Server only.

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

5 **4.6 Data Sets**

6 **Scope:** Client & Server.

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

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

12 **4.6.1 Attributes**

13 — NAME

14 — VALUE

15 — SIZEOF_VALUE

16 **4.7 Client Groups**

17 **Scope:** Server only.

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

20 **4.7.1 Attributes**

21 — TYPE (STATIC or DYNAMIC)

22 — List of CLIENT_SO_GUIDs (only in case of static binding)

23 — List of {PATTERN, ATTRIBUTE} tuple.

24

25

1 **4.8 Key Groups**

2 **Scope:** Server only.

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

5 **4.8.1 Attributes**

6 — TYPE (STATIC or DYNAMIC)

7 — List of CLIENT_SO_GUIDs (only in case of static binding)

8 — List of {PATTERN, DATASET} tuple.

9

1 **5. Key Management Policies**

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

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

6 **5.1 Key Assignment Policy**

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

10 A key assignment policy determines

- 11 — the type of unencrypted data that is determined to be encrypted with a specific key.
 - 12 — how often to generate new keys
- 13

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

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

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

22 container attribute name = "storage_server_name" value="Ireland.com"

23 data attribute name = "financial"

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

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

32

1 **5.2 Retention Policy**

2 **5.2.1 Overview**

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

6 This policy should be superseded by the key life cycle.

7

8 **5.2.2 Attributes**

9 — T_EXPIRATION

10 — T_DISABLE

11 **5.2.3 States**

12 — Created

13 — Assigned

14 — Enforcing

15 — Disabled

16 **5.2.4 Operations**

17 — Add

18 — Associate

19 — Disassociate

20 — Delete

21

1 **5.3 Wrapping Policy**

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

5 INFORMATIVE: This policy when applied at the key level can dictate whether

6 **5.3.1 Attributes**

7 — WRAPPING_TYPE (asymmetric / symmetric / signature)

8 — WRAPPING_MODE (as listed in the key blob section)

9 — CU_ID (*when enforced at a key level, then this allows the key to be locked down to a particular*
10 *client*).

11

12 **5.3.2 States**

13 — Created

14 — Assigned

15 — Enforcing

16 — Disabled

17 **5.3.3 Operations**

18 — Add

19 — Associate

20 — Disassociate

21 — Delete

22

1 **5.4 Audit Policy**

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

3 *TODO: Bob Lockhart to provide new content.*

4 **5.4.1 Attributes**

5 — Operation type

6 — Event type (to trigger, Log, SNMP trap, etc.) Mandatory: Log

7 — Event Dispatch Destination (Local, SNMP, syslog) Mandatory: Local, syslog.

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

10 **5.4.2 States**

11 — Created

12 — Assigned

13 — Enforcing

14 — Disabled

15 **5.4.3 Operations**

16 — Add

17 — Associate

18 — Disassociate

19 — Delete

20

21

22

1 **5.5 Replication Policy**

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

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

7 **5.5.1 Attributes**

8 — List of KM servers

9 **5.5.2 States**

10 — Created

11 — Assigned

12 — Enforcing

13 — Disabled

14 **5.5.3 Operations**

15 — Add

16 — Associate

17 — Disassociate

18 — Delete

19

20

21

1 **5.6 Access/Distribution Policy**

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

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

6 **5.6.1 Attributes**

7 — SO_GUID

8 — Client list

9 — KM server list.

10 **5.6.2 States**

11 — Created

12 — Assigned

13 — Enforcing

14 — Disabled

15 **5.6.3 Operations**

16 — Add

17 — Associate

18 — Disassociate

19 — Delete

20

1 **5.7 Caching Policy**

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

4 Attributes

5 — T_CACHE_INTERVAL

6 **5.7.1 States**

7 — Created

8 — Assigned

9 — Enforcing

10 — Disabled

11 **5.7.2 Operations**

12 — Add

13 — Associate

14 — Disassociate

15 — Delete

16

1 **6. Key Management Operations**

2 **6.1 Authenticate**

3 Scope: KMCS

4 **6.1.1 Overview**

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

8 **6.1.2 Input / Output / Error**

9 — (I): Client

10 — (O): Credentials (If the request type is login and not validation)

11 — (E): E_INVALID_CREDENTIALS

12 — (E): E_UNSUPPORTED_AUTHENTICATION_MODE

13

14 **6.2 Capability Negotiation**

15 Scope: KMCS

16 **6.2.1 Overview**

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

19 **6.2.2 Input / Output / Error**

20 — (I): Client

21 — (I): List of Capability Objects

22 — (O): List of Capability Objects that are supported by the KM server

23 **6.3 Get Server Capabilities**

24 — (I): Client

25 — (O): List of Capability Objects.

26

1 **6.4 Create/Generate Key**

2 Scope: KMCS, KM Console

3 **6.4.1 Overview**

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

6 **6.4.2 Input / Output / Error**

7 — (I): Client

8 — (I): List of Dataset objects.

9 — (O): Key

10 — (E): ...

11 **6.5 Store Key**

12 Scope: KMCS, KM Console

13 **6.5.1 Overview**

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

15 **6.5.2 Input / Output / Error**

16 (I): Client

17 (I): List of Dataset objects.

18 (O): Key_SO_GUID

19 (E)...

20

1 **6.6 Get Key**

2 **6.6.1 Overview**

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

6 **6.6.2 Input / Output / Error**

7 (I): Client

8 (I): List of Dataset objects.

9 (O): Key

10 (E): ...

11 **6.7 Push Audit Message**

12 **6.7.1 Overview**

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

16 **6.7.2 Input / Output / Error**

17 (I): Client

18 (I): Key_SO_GUID

19 (I): Message (Optional)

20 (O): Boolean – SUCCESS/FAILURE.

21 (E): ...

22 **6.8 Get Random Bytes**

23 **6.8.1 Input / Output / Error**

24 — (I): Client

25 — (I): numbers of bytes desired

26 — (O) Base64 Encoded bytes

27 **7. Key Management Transport**

1 *[Supported transport protocols go here]*

2 **8. Key Management Messaging**

3 *[This is an additional section that I am proposing we add to help complete the standard. It does not*
4 *currently exist in Draft 1.]*

5 *[This section would contain normative information for the XML and/or TLV formats we decide to support]*

6

1 **Annex A**

2 (informative)

3 **Bibliography**

4 *[List all bibliographic material here]*

5

1 **Annex B** (informative)

2 **Example Use Cases**

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

4

1 **Annex C** (informative)

2 **XML and TLV Schema Definitions**

3 **C.1 XML Schema**

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

5 **C.2 TLV Schema**

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

7

8