To: IEEE P1619.3 Task Group
From: P1619.3 [OO & Messaging], Members:

Date: March 13, 2008
Purpose: Proposed changes against P1619.3/D2 to incorporate the Objects and Operations Proposal

Matthew Ball 3/13/08 10:33 AM
Deleted: March 10, 2008

Matthew Ball 3/10/08 2:23 PM
Deleted: 1

Matthew Ball 3/10/08 2:23 PM
Deleted: [_____]

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. 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 9-17-07).*

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

Matthew Ball 3/10/08 2:23 PM
Deleted: 1

Changes to P1619.3/D2

[Note any sections that are to be completely removed from the existing document.]
[Note text or verbiage changes in the appropriate sections if the section is not to be fully deleted.]

2. Normative References

[Include any normative references in this section]

3. 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]

3.1 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.]*

3.2 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.]

4. 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]

5. Key Management Objects & Operands

5.1 Overview

Objects defined in this section may be categorized as either server only, client only or both.

Operations defined in this section may be categorized as KMCS, KMSS or KM Admin Interface. KM Console/UI operations listed here are informative and are outside the purview of the standard.

5.2 Key

Scope: Client & Server.

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

5.2.1 Attributes

A key object is defined to consist of the following:

- SO_GUID
- CREATOR_SO_GUID
- Time Stamps (T_CREATED, T_EXPIRATION, T_LAST_STATE_MODIFICATION)
- STATE (Will be one of the states as listed in the architecture specification)
- CIPHER_TYPE (e.g. XTS, AES-256, GCM, Unspecified, etc.) There will be a mandatory set of cipher suites that need to be supported. Additional ciphers may be supported by the KM server. ***Note: add NIST + 1619.x as mandatory cipher types that need to be supported.***
- KEY_BLOB
- KEY_USAGE (type: string)
- KEY_ENCODING (type:string) (e.g., XML base64, ASC hex, binary, unspecified)
- *WRAPPINGKEY_SO_GUID (NOTE: A FEW WRAPPING METHODS SHOULD BE PART OF THE STANDARD)*
- *WRAP_TYPE* – (e.g, NONE, AES Key Wrap, RSA OAEP, ECC ECIES, unspecified, vendor specific)
- *VENDOR_SPECIFIC_EXTENSIONS*
- *APPLICATION_EXTENSIONS*

Matthew Ball 3/10/08 2:38 PM
Formatted: Bullets and Numbering

5.2.2 States

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

5.2.3 Operations

- Create
- Get

— Store

5.3 Client

Scope: Client & Server.

The client object consists of its credentials and capabilities.

5.3.1 Attributes

— SO_GUID
— CREDENTIAL SO_GUID
— CAPABILITY SO_GUID

5.3.2 States

— Active
— Disabled/Locked
— Authenticated

5.3.3 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.

5.4 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.

5.4.1 Attributes (proposed initial set)

— SO_GUID
— NAME

— VALUE
— SIZEOF_VALUE

The standard defines the following pre-defined datasets to ensure inter-operability across KM servers:

— storageType
— storageServerName
— storageShareName
— storagePath
— storageTargetName
— storageInitiatorName
— storagePortWwn
— storageHostWwn
— storageLun
— storagePoolLabel
— storageTapeLabel
— driveId
— serialNumber
— userId
— StartingBlockOffset
— DataLength

Matthew Ball 3/10/08 2:43 PM
Deleted: must

Matthew Ball 3/10/08 2:49 PM
Deleted: support certain

Matthew Ball 3/10/08 2:49 PM
Deleted: . The names of the mandatory datasets are

As the data attributes are defined within each organization, they can be any string. The string must be registered in both the KMS and within the organization. The attribute should be unique within the ID space, i.e. within the set of coordinating Key Management Servers.

Regular expressions used to associate “KeyPools” and “Client Groups” are to be used to

5.5 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.

5.6 Capability

5.6.1 Attributes

— SO_GUID

— MESSAGING_TYPE (BITMAP - SOAP / ASN.1-DER)

5.6.2 States

None.

5.6.3 Operations

No direct operations. The capability object is sent as part of the session initiation message. If a KM server cannot support the clients capabilities, then the session is rejected, else the KM server picks one from the list of capabilities at its discretion.

5.7 Credentials

The credentials object is presented to the KM for verification of client authenticity. This object might also be returned to the client as part of a login request.

5.7.1 Attributes

— SO_GUID

— Credential Type (Session, Username/Password, Asymmetric/symmetric key, SSO, ***CHAP/AKEP2?***)

NOTE: need to have a policy to restrict credential types.

5.7.2 States

None.

5.7.3 Operations

None directly. A credential object is passed in every request that is made by a client.

5.8 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.

5.8.1 Attributes

— SO_GUID

— TYPE (STATIC or DYNAMIC)

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

— List of {PATTERN, ATTRIBUTE} tuple.

5.9 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.

5.9.1 Attributes

— SO_GUID

— TYPE (STATIC or DYNAMIC)

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

— List of {PATTERN, DATASET} tuple.

6. 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.

6.1 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.

6.2 Retention Policy

Scope: Server only.

6.2.1 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.

6.2.2 Attributes

— SO_GUID

— T_EXPIRATION

— T_DISABLE

6.2.3 States

— Created

— Assigned

— Enforcing

— Disabled

6.2.4 Operations

— Add

— Associate

— Disassociate

— Delete

6.3 Wrapping Policy

Scope: Server only.

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.

6.3.1 Attributes

— SO_GUID

— WRAPPING_TYPE (asymmetric / symmetric / signature)

— WRAPPING_MODE (list various modes – ***TODO)***

6.3.2 States

— Created

— Assigned

— Enforcing

— Disabled

6.3.3 Operations

— Add

— Associate

— Disassociate

— Delete

6.4 Audit Policy

Scope: Server only.

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

TODO: Bob Lockhart to provide new content.

6.4.1 Attributes

— SO_GUID

— 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.

6.4.2 States

— Created

— Assigned

— Enforcing

— Disabled

6.4.3 Operations

— Add

— Associate

— Disassociate

— Delete

6.5 Replication Policy

Scope: Server only.

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.

6.5.1 Attributes

— SO_GUID

— List of KM servers

6.5.2 States

— Created

— Assigned

— Enforcing

— Disabled

6.5.3 Operations

— Add

— Associate

— Disassociate

— Delete

6.6 Access/Distribution Policy

Scope: Server only.

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.

6.6.1 Attributes

— SO_GUID
— Client list
— KM server list.

6.6.2 States

— Created
— Assigned
— Enforcing
— Disabled

6.6.3 Operations

— Add
— Associate
— Disassociate
— Delete

7. Key Management Operations

7.1 Protocol/Capability Negotiation

Scope: KMCS

7.1.1 Overview

The client sends its capabilities to the server and the server picks one from the list based on its implementation. If the server is unable or unwilling to accept the capabilities as presented by the client, it rejects the request.

7.1.2 Input / Output / Error

- (I): Capability
- (O): Capability (as chosen by the KM server)
- (E): E_CAPABILITY_UNSUPPORTED.

7.2 Authenticate

Scope: KMCS

7.2.1 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/IPSec) or at messaging level. Every request must still contain the "authentication" object with the credential_type set so that the KM server can validate the channel security attributes.

7.2.2 Input / Output / Error

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

7.3 Create/Generate Key

Scope: KMCS, KM Console

7.3.1 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.

7.3.2 Input / Output / Error

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

7.4 Store Key

Scope: KMCS, KM Console

7.4.1 Overview

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

7.4.2 Input / Output / Error

- (I): Client_SO_GUID
- (I): Authentication
- (I): List of Dataset objects.
- (O): Key_SO_GUID
- (E)…

7.5 Get Key

Scope: KMCS

7.5.1 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.

7.5.2 Input / Output / Error

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

7.6 Modify Key (attributes)

Scope: KM Console

7.6.1 Overview

Authorized clients are allowed to modify key attributes/meta-data. The actual key can never be modified.

Discussion: State modification – what is the end result of our discussion.

7.7 Create KeyPool

Scope: KM Console

7.7.1 Overview

A KeyPool is grouping of keys. The grouping can be static (members have to be manually added) or dynamic (based on a pattern in either key names, data sets, etc..)

7.7.2 Input / Output / Error

- (I): KeyPool_SO_GUID
- (I) Pool Type (Static or Dynamic)
- (I) {Pattern, Dataset.Name} tuples

— (E): …

7.8 Add Keys to KeyPool

Scope: KM Console

7.8.1 Overview

This operation permits an administrator with authority to add keys to a KeyPool.

7.8.2 Input / Output / Error

— (I) List of KeyPool_SO_GUID if static
— (I) {Pattern, Dataset.Name} tuples if dynamic
— (I) KeyPool_SO_GUID
— (O) Boolean that indicates Success or Failure
— (E) …

7.9 Remove Keys From KeyPool

Scope: KM Console

7.9.1 Overview

This operation permits clients with authority to remove keys from a KeyPool.

7.9.2 Input / Output / Error

— (I) List of KeyPool_SO_GUID
— (I) Key_SO_GUID
— (O) Boolean that indicates Success or Failure
— (E) …

7.10 Join/Merge ‘n’ KeyPools.

Scope: KM Console

7.10.1 Overview

This operation permits clients to merge KeyPools.

7.10.2 Input / Output / Error

— (I) List of KeyPool_SO_GUID if static
— (I) {Pattern, Dataset.Name} tuples if dynamic
— (O) KeyPool_SO_GUID
— (E) …

7.11 Split KeyPools based on a pattern (regex)

Scope: KM Console

7.11.1 Overview

Splits a given keypool into multiple keypools based on a pattern.

7.11.2 Input / Output / Error

— (I) KeyPool_SO_GUID
— (I) {Pattern, Dataset.Name} tuples if dynamic
— (O) Pair of KeyPool_SO_GUID
— (E) …

7.12 Create Client Group

Scope: KM Console

7.12.1 Overview

This operation allows a client to create client groups.

7.12.2 Input / Output / Error

— (I) Client Group Name
— (I) Group Type (Static / Dynamic)
— (I) {Pattern,ClientAttributeName} tuples if dynamic
— (O) Client_Group_SO_GUID
— (E) …

7.13 Add/Remove client to/from Client Group

Scope: KM Console

7.13.1 Overview

This operation removes a client(s) from a client group.

7.13.2 Input / Output / Error

- (I) List of Client_SO_GUID if the binding is static.
- (I) {Pattern, ClientAttributeName } tuples if the binding is dynamic.

7.14 Push Audit Message

7.14.1 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.

7.14.2 Input / Output / Error

- (I): Client_SO_GUID
- (I): Authentication
- (I): Key_SO_GUID
- (I): Message (Optional)
- (O): Boolean – SUCCESS/FAILURE.
- (E): …

8. Key Management Transport

TODO

9. Key Management Messaging

TODO

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]