An Inside Look at Imminent Key Management Standards

Matt Ball, Oracle Corporation
The material contained in this tutorial is copyrighted by the SNIA.

Member companies and individual members may use this material in presentations and literature under the following conditions:
- Any slide or slides used must be reproduced in their entirety without modification
- The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.

This presentation is a project of the SNIA Education Committee.

Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.

The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.
Abstract

An Inside Look at Key Management Standards

This session provides storage managers and planners an inside look at the expected timing of publishing emerging key management standards, as well as the direction that these standards have taken and the technologies they use. Attendees will learn about existing standards, including OASIS KMIP and IEEE P1619.3, and the implications of the minimum requirements for compliance with each.

For example, the first draft of OASIS KMIP provides a good basic set of key management objects and operations, but leaves out some of the trickier aspects, such as enrollment and discovery. IEEE P1619.3 intends to augment OASIS KMIP by adding support for these features, among others.
Learning Objectives

- Discover the background of existing key management standards, including OASIS KMIP and IEEE P1619.3
- Learn the enabling technologies behind these key management standards
- Apply these standards to integration with existing systems or creation of new systems
Outline

- Motivation for key management
- Overview of key management standards
  - ISO 11770, NIST SP 800-57 part 1-3, OASIS EKMI, IETF KEYPROV, OASIS KMIP, IEEE P1619.3
- Details of OASIS KMIP – Objects, Attributes, Operations, Profiles
- IEEE P1619.3 enhancements to KMIP
- Summary/Questions
Motivation for Key Management
Motivation – Why Key Management?

- Government legislation is driving the requirement to encrypt confidential data
  - Sarbanes-Oxley (SOX), California SB-1386 (July 2003), Health Insurance Portability and Accountability Act (HIPAA), Gramm-Leach-Bliley Act, Payment Card Industry standards (PCI), EU Data Protection Directives

- Customers are demanding that key management vendors provide interoperability to avoid lock-in.

Check out SNIA Tutorial: Introduction to Storage Security
Typical Enterprise Key Management Requirements

- Securely deliver encryption keys to authorized users
- Fail-over and disaster recovery for key management servers
- Maintain audit records to show compliance
- Ease-of-use to manage users (e.g., LDAP)
- Interoperability with other servers and devices
- One use per key (e.g., signing vs. encrypting)
Key Management For Storage

Check out SNIA Tutorial:
An Introduction to Key Management for Storage Security

An Inside Look at Imminent Key Management Standards
© 2010 Storage Networking Industry Association. All Rights Reserved.
Overview of Key Management Standards
Standards Timeline History

1996 ... 2006 2007 2008 2009 2010

Frameworks:
- NIST SP 800-57 Part 1, 2 (Mar 07)
- NIST SP 800-57 Part 3

Protocols:
- OASIS EKMI
- IETF KEYPROV
- IEEE P1619.3
- OASIS KMIP
Details of OASIS KMIP
OASIS KMIP Overview

Objects:
- Base Objects
- Managed Objects

Attributes

Client-to-Server Operations

Server-to-Client Operations

Message Contents and Format

Message Encoding

Conformance
KMIP Base Objects

- Attribute
- Credential (e.g., Username & Password)
- Key Block
- Key Value
- Key Wrapping Data
- Key Wrapping Specification
- Transparent Key Structures
- Template-Attribute Structures
KMIP Managed Objects

- Certificate (e.g., X509, PGP)
- Keys:
  - Symmetric Key
  - Public Key
  - Private Key
  - Split Key
- Template
- Secret Data
- Opaque Object
KMIP Attributes (Page 1)

- Unique Identifier
- Name
- Object Type
- Cryptographic Algorithm/Length/Parameters
- Certificate Type/Identifier/Subject/Issuer
- Digest
- Operation Policy Name
- Cryptographic Usage Mask
- Lease Time
- Usage Limits
- State
KMIP Attributes (Page 2)

- Date: Initial / Activation / Process Start / Protect Stop / Deactivation / Destroy / Compromise Occurrence / Compromise
- Revocation Reason
- Archive Date
- Object Group
- Link
- Application Specific Information
- Contact Information
- Last Change Date
- Custom Attribute
NIST Key Lifecycle Model

- Pre-Activation
- Active
- Deactivated
- Compromised
- Destroyed
- Destroyed Compromised
KMIP Client-to-Server Operations

- Create
- Create Key Pair
- Register
- Re-key
- Derive Key
- Certify
- Re-certify
- Locate
- Check
- Get (attributes (list))

- Add/Modify/Delete Attribute
- Obtain Lease
- Get Usage Allocation
- Activate
- Revoke
- Destroy
- Archive
- Recover
- Validate
- Query/Cancel/Poll
KMIP Message Encoding TTLV

- TTLV = Tag, Type, Length, Value
- Chosen to be simple to process with 32-bit/64-bit embedded processors
- Types:
  - Integer, Long Integer, Big Integer
  - Enumeration
  - Boolean
  - Text String, Byte String
  - Date-Time
  - Interval
  - Structure
KMIP TTLV Examples

- An Integer containing the decimal value 8: 1699
  - 42 00 20 | 02 | 00 00 00 04 | 00 00 00 08 00 00 00 00

- A Long Integer containing the decimal value 123456789000000000:
  - 42 00 20 | 03 | 00 00 00 08 | 01 B6 9B 4B A5 74 92 00

- A Big Integer containing the decimal value 1234567890000000000000000000:
  - 42 00 20 | 04 | 00 00 00 10 | 00 00 00 00 03 FD 35 EB 6B C2 DF 46 18 08 00 00

- An Enumeration with value 255:
  - 42 00 20 | 05 | 00 00 00 04 | 00 00 00 FF 00 00 00 00

- A Boolean with the value True:
  - 42 00 20 | 06 | 00 00 00 08 | 00 00 00 00 00 00 00 01
KMIP TTLV Examples (Cont)

- A Text String with the value "Hello World":
  ```
  42 00 20 | 07 | 00 00 00 0B | 48 65 6C 6F 20 57 6F 72 6C 64 00 00 00 00
  ```

- A Byte String with the value \{ 0x01, 0x02, 0x03 \}:
  ```
  42 00 20 | 08 | 00 00 00 03 | 01 02 03 00 00 00 00 00
  ```

- A Date-Time, containing the value for Friday, March 14, 2008, 11:56:40 GMT:
  ```
  42 00 20 | 09 | 00 00 00 08 | 00 00 00 00 47 DA 67 F8
  ```

- An Interval, containing the value for 10 days:
  ```
  42 00 20 | 0A | 00 00 00 04 | 00 0D 2F 00 00 00 00 00
  ```

- A Structure containing an Enumeration, value 254, followed by an Integer, value 255, having tags 420004 and 420005 respectively:
  ```
  42 00 20 | 01 | 00 00 00 20 | 42 00 04 | 05 | 00 00 00 04 | 00 00 00 FE
  00 00 00 00 | 42 00 05 | 02 | 00 00 00 04 | 00 00 00 FF 00 00 00 00
  ```
KMIP Request Message Format

- **Request Header**
  - Protocol Version
  - Maximum Response Size (optional)
  - Asynchronous Indicator (optional)
  - Authentication (optional)
  - ...
  - Batch Count

- **Batch Item (one or more)**
  - Operation
  - Unique Batch Item ID (optional)
  - Request Payload
KMIP Response Message Format

- **Response Header**
  - Protocol Version
  - Time Stamp
  - Batch Count

- **Batch Item (one or more)**
  - Operation
  - Unique Batch Item ID
  - Result Status/Reason/Message
  - Asynchronous Correlation Value
  - Response Payload
  - Message Extension
KMIP Create Key Request/Response

Create Key Request

Request Message

Request Header
- Protocol Version: 1.0
- Batch Count: 1

Batch Item
- Operation: Create

Request Payload
- Object Type: Symmetric Key
- Template-Attribute
  - Name: Template1
- Attribute: Crypto Algorithm: AES
- Attribute: Crypto Length: 128
- Attribute: Crypto Usage Mask: E/D

Create Key Response

Response Message

Request Header
- Protocol Version: 1.0

Batch Item
- Operation: Create

Response Payload
- Object Type: Symmetric Key
- Unique Identifier: 61b10614-d8b5-...
**KMIP Create Key Request**

**Request:**

Create (symmetric key using template)

In: `objectType='00000002', template={ 'NameValue='Template1', NameType='00000001' }, attributes={ 'CryptographicAlgorithm='AES', CryptographicLength='128', CryptographicUsageMask='0000000C' }`

<table>
<thead>
<tr>
<th>Tag</th>
<th>Data Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Message</td>
<td><code>0x420078</code>, Type: Structure (0x01), Data:</td>
</tr>
<tr>
<td>Request Header</td>
<td><code>0x420077</code>, Type: Structure (0x01), Data:</td>
</tr>
<tr>
<td>Protocol Version</td>
<td><code>0x420069</code>, Type: Structure (0x01), Data:</td>
</tr>
<tr>
<td>Protocol Version Major</td>
<td><code>0x42006A</code>, Type: Integer (0x02), Data: 0x00000001 (1)</td>
</tr>
<tr>
<td>Protocol Version Minor</td>
<td><code>0x42006B</code>, Type: Integer (0x02), Data: 0x00000000 (0)</td>
</tr>
<tr>
<td>Batch Count</td>
<td><code>0x42000D</code>, Type: Integer (0x02), Data: 0x00000001 (1)</td>
</tr>
<tr>
<td>Batch Item</td>
<td><code>0x42000F</code>, Type: Structure (0x01), Data:</td>
</tr>
<tr>
<td>Operation</td>
<td><code>0x42005C</code>, Type: Enumeration (0x05), Data: 0x00000001 (Create)</td>
</tr>
<tr>
<td>Request Payload</td>
<td><code>0x420079</code>, Type: Structure (0x01), Data:</td>
</tr>
<tr>
<td>Object Type</td>
<td><code>0x420057</code>, Type: Enumeration (0x05), Data: 0x00000002 (Symmetric Key)</td>
</tr>
<tr>
<td>Template-Attribute</td>
<td><code>0x420091</code>, Type: Structure (0x01), Data:</td>
</tr>
<tr>
<td>Name</td>
<td><code>0x420053</code>, Type: Structure (0x01), Data:</td>
</tr>
<tr>
<td>Name Value</td>
<td><code>0x420055</code>, Type: Text String (0x07), Data: Template1</td>
</tr>
<tr>
<td>Name Type</td>
<td><code>0x420054</code>, Type: Enumeration (0x05), Data: 0x00000001 (Uninterpreted text string)</td>
</tr>
<tr>
<td>Attribute</td>
<td><code>0x420008</code>, Type: Structure (0x01), Data:</td>
</tr>
<tr>
<td>Attribute Name</td>
<td><code>0x42000A</code>, Type: Text String (0x07), Data: Cryptographic Algorithm</td>
</tr>
<tr>
<td>Attribute Value</td>
<td><code>0x42000B</code>, Type: Enumeration (0x05), Data: 0x00000003 (AES)</td>
</tr>
<tr>
<td>Attribute</td>
<td><code>0x420008</code>, Type: Structure (0x01), Data:</td>
</tr>
<tr>
<td>Attribute Name</td>
<td><code>0x42000A</code>, Type: Text String (0x07), Data: Cryptographic Length</td>
</tr>
<tr>
<td>Attribute Value</td>
<td><code>0x42000B</code>, Type: Integer (0x02), Data: 0x00000080 (128)</td>
</tr>
<tr>
<td>Attribute</td>
<td><code>0x420008</code>, Type: Structure (0x01), Data:</td>
</tr>
<tr>
<td>Attribute Name</td>
<td><code>0x42000A</code>, Type: Text String (0x07), Data: Cryptographic Usage Mask</td>
</tr>
<tr>
<td>Attribute Value</td>
<td><code>0x42000B</code>, Type: Integer (0x02), Data: 0x00000000C (Encrypt, Decrypt)</td>
</tr>
</tbody>
</table>
Out: objectType='00000002', uuidKey

Tag: Response Message (0x42007B), Type: Structure (0x01), Data:
  Tag: Response Header (0x42007A), Type: Structure (0x01), Data:
    Tag: Protocol Version (0x420069), Type: Structure (0x01), Data:
      Tag: Protocol Version Major (0x42006A), Type: Integer (0x02), Data: 0x00000001 (1)
      Tag: Protocol Version Minor (0x42006B), Type: Integer (0x02), Data: 0x00000000 (0)
    Tag: Time Stamp (0x420092), Type: Date-Time (0x09), Data: 0x000000004AFBE7C5 (Thu Nov 12 11:47:33 CET 2009)
    Tag: Batch Count (0x42000D), Type: Integer (0x02), Data: 0x00000001 (1)
  Tag: Batch Item (0x42000F), Type: Structure (0x01), Data:
    Tag: Operation (0x42005C), Type: Enumeration (0x05), Data: 0x00000001 (Create)
    Tag: Result Status (0x42007F), Type: Enumeration (0x05), Data: 0x00000000 (Success)
  Tag: Response Payload (0x42007C), Type: Structure (0x01), Data:
    Tag: Object Type (0x420057), Type: Enumeration (0x05), Data: 0x00000002 (Symmetric Key)
    Tag: Unique Identifier (0x420094), Type: Text String (0x07), Data: 61b10614-d8b5-46f9-8d17-2fa6ea1d747a
KMIP Conformance Profiles

- (Based on second public review of v1.0 draft)
- To claim conformance, a "KMIP Server" must support at least one of these profiles:
  - Secret Data Server
  - Basic Symmetric Key Store and Server
  - Basic Symmetric Key Foundry and Server
- Must also support an https authentication suite:
  - Basic Authentication Suite: TLS 1.0, or
  - TLS 1.2 Authentication Suite
- Must support 3DES and AES Keys
OASIS KMIP Tools

- Roll your own! (for handling the binary)
  - No publicly available KMIP implementations yet
  - Interop demo at the 2010 RSA Conference
- For TLS encryption, could use OpenSSL
- The usual Linux and Windows tools still apply:
  - Linux: Apache, Tomcat, etc
  - Windows: Internet Information Server (IIS), etc
Details of IEEE P1619.3
Using P1619.3 extensions to KMIP

- Standard in progress, so details may change!
- Defines a RESTful and SOAP-based web service using WSDL 2.0.
  - RESTful interface is based on resources (objects)
  - SOAP is based on procedure calls (actions)
- Map KMIP binary types to XML types
- Many tools exist to help with XML processing
- Keep the same object model as KMIP so that the back-end database can be the same.
REST Summary

▶ Stands for REpresentational State Transfer
▶ Introduced by Roy Fielding in 2000 thesis
▶ A web service that follows the REST guidelines is said to be "RESTful"
▶ Uses HTTP/1.1 commands:
  ◆ GET – Retrieve a resource without side-effects
    › Allows for caching
  ◆ PUT – Update or create a resource or collection
  ◆ POST – Issue command with potential side-effects, or create a new resource or collection
  ◆ DELETE – Remove a resource or collection
A WSDL (pronounced Wiz-Dull) is an XML-based schema for describing the objects and operations of a web service.

- WSDL 1.1 was originally for describing SOAP.
- WSDL version 2.0 was published in June 2007.
- New features include support for RESTful bindings.
- Some tools support WSDL 2.0, including Apache AXIS2 (for both Java and C).
### Comparison of Primitive Types

<table>
<thead>
<tr>
<th>KMIP Primitive Type</th>
<th>P1619.3 XML Type</th>
<th>Typical C++ Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>xsd:int</td>
<td>int (32-bit)</td>
</tr>
<tr>
<td>Long Integer</td>
<td>xsd:long</td>
<td>long long (64-bit)</td>
</tr>
<tr>
<td>Big Integer</td>
<td>xsd:base64Binary</td>
<td>struct {}</td>
</tr>
<tr>
<td>Enumeration</td>
<td>xsd:string</td>
<td>enum { ... }</td>
</tr>
<tr>
<td>Boolean</td>
<td>xsd:boolean</td>
<td>bool</td>
</tr>
<tr>
<td>Text String</td>
<td>xsd:string</td>
<td>wchar_t *</td>
</tr>
<tr>
<td>Byte String</td>
<td>xsd:base64Binary</td>
<td>struct {}</td>
</tr>
<tr>
<td>Date-Time</td>
<td>xsd:dateTime</td>
<td>time_t (64-bit)</td>
</tr>
<tr>
<td>Interval</td>
<td>xsd:duration</td>
<td>char * (or long long)</td>
</tr>
</tbody>
</table>
POST /P1619-3-KMIP/Create HTTP/1.1
Host: www.example.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Accept: application/xml
X-P1619-3-KMIP-Version: 1.0

<ObjectType>SymmetricKey</ObjectType>
<TemplateAttribute>
  <NameList><item>
    <NameValue>Template1</NameValue>
    <NameType>UninterpretedTextString</NameType>
  </item></NameList>
  <AttributeList><item>
    <AttributeName>CryptographicAlgorithm</AttributeName>
    <AttributeValue>
      <Enumeration><CryptographicAlgorithm>AES</CryptographicAlgorithm></Enumeration>
    </AttributeValue>
  </item><item>
    <AttributeName>CryptographicLength</AttributeName>
    <AttributeValue><Integer>128</Integer></AttributeValue>
  </item><item>
    <AttributeName>CryptographicUsageMask</AttributeName>
    <AttributeValue><Enumeration><CryptographicUsageMask>Encrypt</CryptographicUsageMask><CryptographicUsageMask>Decrypt</CryptographicUsageMask></Enumeration></AttributeValue>
  </item></AttributeList>
</TemplateAttribute>
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnn
Date: Tue, 15 Nov 2009 08:12:31 GMT

<ObjectType>SymmetricKey</ObjectType>
<UniqueIdentifier>61b10614-d8b5-46f9-8d17-2fa6ealid747a</UniqueIdentifier>
Can pass parameters as IRI-encoded URL

- Example: GET /get?UniqueIdentifier=abc

Can also access directories with UIDs

- Example: GET /Objects/abc

Example:

GET /P1619-3-KMIP/get?UniqueIdentifier=abc
Host: www.example.com
Accept: application/xml
P1619-3-KMIP-Version: 1.0

HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnn
Date: Tue, 15 Nov 2009 08:12:31 GMT

<ObjectType>Certificate</ObjectType> ...
P1619.3 RESTful services

- Operations mapped as URIs
- Managed objects optionally mapped as URIs:
  - /Objects/{UniqueIdentifier}
- Attributes as a proper of particular objects:
  - /Objects/{UniqueIdentifier}/Attributes/{AttributeName}
- Can use HTTP GET to view objects or attributes, allowing the server to cache the results
P1619.3 Cluster Discovery

- P1619.3 Discovery service allows client to discover other available servers in the cluster.
- Provides way for client to perform automatic failover if primary server is unavailable.
- Details are still being worked out by the P1619.3 task group.
- Many of these changes will likely be integrated into future KMIP specifications or profiles.
P1619.3 will also provide a way to perform a client enrollment operation, using some form of credentials (like username/password, token, etc).

Details are being worked on within the P1619.3 task group.

Similar changes are under consideration for KMIP v1.1.
P1619.3 Implementation Tools

- Many tools that convert WSDL to source code
- gSOAP for C/C++
  - See [http://www.cs.fsu.edu/~engelen/soap.html](http://www.cs.fsu.edu/~engelen/soap.html)
- Apache AXIS2 (available for both Java and C)
- Apache CXF
- Many more!
<table>
<thead>
<tr>
<th>Attribute</th>
<th>OASIS KMIP</th>
<th>IEEE P1619.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall Format</td>
<td>TTLV Binary</td>
<td>RESTful XML</td>
</tr>
<tr>
<td>Message Version</td>
<td>Version field in header</td>
<td>HTTP header version</td>
</tr>
<tr>
<td>Error Reporting</td>
<td>Error field in header</td>
<td>HTTP Error codes</td>
</tr>
<tr>
<td>Tool Support</td>
<td>No tool support</td>
<td>Many XML libraries</td>
</tr>
<tr>
<td>Grammar Validation</td>
<td>Manual</td>
<td>WSDL validation</td>
</tr>
<tr>
<td>Code size</td>
<td>Small</td>
<td>Medium</td>
</tr>
<tr>
<td>Processing Overhead</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Security</td>
<td>HTTPS with TLS</td>
<td>HTTPS with TLS</td>
</tr>
<tr>
<td>HTTP Command</td>
<td>POST</td>
<td>GET, POST, DELETE</td>
</tr>
<tr>
<td>Asynchronous Msg</td>
<td>Async flag in header</td>
<td>POST to startCmd</td>
</tr>
<tr>
<td>Command Batching</td>
<td>Count in header</td>
<td>Keep session alive</td>
</tr>
</tbody>
</table>
Projected Timeline

**OASIS KMIP:**
- 1st Public Review:
- 2nd Public Review:
- Publication:

**IEEE P1619.3:**
- XML Mapping:
- Enrollment/Discovery:
- Working Group Ballot:
- Sponsor Ballot:
For more information

- **OASIS KMIP Homepage:**
  - [http://www.oasis-open.org/committees/kmip](http://www.oasis-open.org/committees/kmip)

- **IEEE P1619 Security in Storage Working Group:**
  - [http://siswg.net](http://siswg.net)

- **IEEE Key Management Summit 2010**
  - May 4-5, 2010 at Lake Tahoe, NV
Q&A / Feedback

Please send any questions or comments on this presentation to SNIA: tracksecurity@snia.org

Many thanks to the following individuals for their contributions to this tutorial.
- SNIA Education Committee

Gianna DaGiau
Mark Carlson
Larry Hofer, CISSP, PE