Multi-Vendor Key Management with KMIP

Tim Hudson
CTO & Technical Director
CRYPTSOFT
tjh@cryptsoft.com
Abstract

- Practical experience from implementing KMIP and from deploying and interoperability testing multiple vendor implementations of KMIP.
- Guidance covering the key issues you need to ensure that your vendors address
- How to distinguish between simple vendor tick-box approaches to standard conformance and actual interoperable solutions.
The need for

KEY MANAGEMENT
Why Key Management

- Encryption is the primary means of securing stored data
- Data Encryption without adequate key management is pointless
Encryption impact on Data Protection

- **Data Protection (Storage)**
  
  *Assurance that data is not corrupted, is accessible for authorized purposes only, and is in compliance with applicable requirements.*

  - Storage Networking Industry Association Dictionary

- **Data Protection (Security)**
  
  *The implementation of appropriate administrative, technical or physical means to guard against unauthorized intentional or accidental disclosure, modification, or destruction of data.*

  - ISO/IEC 2382-1:1993

Source: Eric Hibbard – Hitachi Data Systems
Why Key Management Standards

- Standards are of limited use if not implemented properly
- Standards are also of limited use if they are not widely adopted
Key Management Standards

- NSA EKMS
- OASIS EKMI
- ANSI X9.24
- IEEE P1619.3
- OASIS KMIP
- IETF KEYPROV

- NIST SP 800-57
- NIST SP 800-130
- NIST SP 800-152
- ISO 11770
Specifications & Standards

- ISO/IEC
  - ISO/IEC 11770
  - ISO/IEC 27040, ISO/IEC 18033
- NIST
  - NIST FIPS 140-2, NIST SP 800-57
  - NIST SP 800-130, NIST SP 800-152
  - NIST SP 800-88
- OASIS
  - KMIP (Key Management Interoperability Protocol)
FIPS 140-2 Key Management

Requirements

Count

Area

1 2 3 4 5 6 7 8 9 10 11 12 13 14

physical
self-tests
key mgmt
spec
ports+i/f
op env
FSM
emi/emc
design
mitigation
docs
dev
secpol

TE
VE
AS
NIST SP 800-130 CKMS Framework Requirements

- Count
- Area
  - intro
  - basics
  - goals
  - policies
  - roles
  - controls
  - testing
  - assessment
  - DR
  - tech chal
  - interop
  - keys & metadata

8/321
NIST SP 800-152 Federal KM Profile

NIST SP 800-152 Federal CKMS Requirements, Augmentations, and Features

Count

Area

intro basics goals roles interop controls testing DR assessment tech chal

PF PA PR

keys & metadata

23/369
The need for multi-vendor KEY MANAGEMENT
Multi-Vendor – Single Integration

Prior to a standard each application had to support each vendor protocol

With a standard each application only requires support for one protocol
Multi-Vendor – Single Integration

- Positive
  - Single Integration with single SDK
  - Common vocabulary
  - Greater choice of technology providers
  - “Free” interoperability without point-to-point testing

- Negative
  - Must follow a standard
  - Vocabulary may not match current usage
  - May need to implement more than is strictly necessary
  - No control over end-user integration
Real-world usage of OASIS KMIP

KEY MANAGEMENT
What is KMIP?

- Key Management Interoperability Protocol
  - “The OASIS KMIP TC works to define a single, comprehensive protocol for communication between encryption systems and a broad range of new and legacy enterprise applications, including email, databases, and storage devices. By removing redundant, incompatible key management processes, KMIP will provide better data security while at the same time reducing expenditures on multiple products.” - https://www.oasis-open.org-committees/tc_home.php?wg_abbrev=kmip
  - A protocol for enterprise management of “stuff”

- OASIS KMIP TC Membership (foundational and sponsor)
  - Cryptsoft, Dell, EMC, Fornetix, Futurex, Hancom Secure, Hewlett Packard Enterprise, IBM, NetApp, Oracle, SafeNet, Symantec, VMware, Vormetric
KMIP Specification History

2007
- SKMP renamed Key Management Interoperability Protocol (KMIP)
- Moved to OASIS as the KMIP Technical Committee
- Standard Key Management Protocol (SKMP) specification formed by private industry group

2009
- KMIP v1.0 OASIS Specification

2010
- KMIP v1.1 OASIS Specification
- KMIP v1.2 Committee Draft
- KMIP v1.2 OASIS Specification Final Committee Draft
- KMIP Interoperability Demo – RSA 2010
  - HP, IBM, Safenet
- KMIP Interoperability Demonstration – RSA 2011
  - Cryptsoft, Emulex, HDD, HP, IBM, RSA/EMC, Safenet
- KMIP Interoperability Demonstration – RSA 2012
  - Cryptsoft, IBM, NetApp, QLabs, Safenet, Thales
- KMIP Interoperability Demonstration – RSA 2013
  - Cryptsoft, HP, IBM, QLabs, Townsend, Thales, Vormetric

2011
- KMIP v1.2 Scope Agreed

2012
- KMIP v1.1 OASIS Specification
- KMIP v1.2 Committee Draft
- KMIP Interoperability Demonstration – RSA 2012
  - Cryptsoft, Dell, HP, IBM, P6R, Safenet, Thales, Vormetric

2013
- KMIP v1.2 OASIS Specification
- KMIP v1.3 Committee Draft

2014
- KMIP v1.1 OASIS Specification
- KMIP v1.3 Committee Draft
- KMIP Interoperability Demonstration – RSA 2014
  - Cryptsoft, Dell, HP, IBM, P6R, Safenet, Thales, Vormetric

2015
- KMIP v1.2 OASIS Specification
- KMIP v1.2 OASIS Specification
- KMIP v1.3 Public Review
- KMIP v1.4 Interop

2016
- KMIP v1.3 Public Review
- KMIP v1.4 Interop
- KMIP Technical Committee Face-to-Face
- KMIP Interoperability Demonstration – RSA 2016
  - Cryptsoft, HPE, IBM, P6R, Fornetix, Utimaco, Townsend, QLabs
- KMIP Interoperability Demonstration – RSA 2015
  - Cryptsoft, Dell, HP, IBM, P6R, Fornetix, Thales, Vormetric
- KMIP Interoperability Demonstration – RSA 2014
  - Cryptsoft, Dell, HP, IBM, P6R, Safenet, Thales, Vormetric
- KMIP Interoperability Demonstration – RSA 2013
  - Cryptsoft, HP, IBM, QLabs, Townsend, Thales, Vormetric
- KMIP Interoperability Demonstration – RSA 2012
  - Cryptsoft, IBM, NetApp, QLabs, Safenet, Thales
- KMIP Interoperability Demonstration – RSA 2011
  - Cryptsoft, Emulex, HDD, HP, IBM, RSA/EMC, Safenet
- KMIP Interoperability Demo – RSA 2010
  - HP, IBM, Safenet
Multi-Vendor – How many products

KMIP Adoption

KMIP Adoption by Market
KMIP – Adoption (Storage)

- KMIP is present in the following:
  - Device-level
    - Disk arrays
    - Tape libraries
    - Virtual tape libraries
    - Flash storage arrays
    - Storage controllers
    - Storage operating systems
  - Network-level
    - Encrypting switches
  - File/Object-level
    - NAS appliances
  - Application-level

Source: ISO/IEC 27040 - Information technology - Security techniques - Storage security
## Multi-Vendor – Who and Where

### Storage
- Disk Arrays, Flash Storage Arrays, NAS Appliances
- Tape Libraries, Virtual Tape Libraries
- Encrypting Switches
- Storage Key Managers
- Storage Controllers
- Storage Operating Systems

### Infrastructure and Security
- Key Managers
- Hardware security modules
- Encryption Gateways
- Virtualization Managers
- Virtual Storage Controllers
- Network Computing Appliances

### Cloud
- Key Managers
- Compliance Platforms
- Information Managers
- Enterprise Gateways and Security
- Enterprise Authentication
- Endpoint Security

![Vendor Logos]
Multi-Vendor – What

- Disk Arrays, Flash Storage Arrays, NAS Appliances, Storage Operating Systems
  - Vaulting master authentication key
  - Cluster-wide sharing of configuration settings
  - Specific Usage Limits checking (policy)
  - FIPS140-2 external key generation (create, retrieve)
  - Multi-version key support during Rekey
  - Backup and recovery of device specific key sets
Multi-Vendor – What

- Tape Libraries, Virtual Tape Libraries
  - External key generation (create, retrieve)
  - FIPS140-2 external key generation (create, retrieve)
  - Multi-version key support during Rekey

- Encrypting Switches, Storage Controllers
  - Vaulting device or port specific encryption keys
  - Cluster-wide sharing of configuration settings
  - Specific Usage Limits checking (policy)
OASIS KMIP

SPECIFICATION
KMIP Specification History

- **OASIS KMIP 1.0** – PR Nov 2009, CS Jun 2010, OS Oct 2010
  - Specification: 105 pages
  - Profiles: 16 pages
  - Usage Guide: 44 pages
  - Use Cases (Test Cases): 168 pages

  - Specification: 164 pages (+56%)
  - Profiles: 39 pages (+143%)
  - Usage Guide: 63 pages (+43%)
  - Test Cases: 513 pages (+205%)

  - Specification: 188 pages (+14%)
  - Profles (multiple): 871 pages (+2133%)
  - Usage Guide: 78 pages (+24%)
  - Test Cases: 880 pages (+70%)
  - Use Cases: 130 pages
OASIS KMIP - Protocol Concepts

Core Concepts

- **Base Objects**
  - Protocol building blocks and parameter encoding
- **Managed Objects**
  - Core concepts managed by KMIP
  - Cryptographic Managed Objects (objects with key material)
- **Attributes**
  - Details related to or about a managed object
- **Client-to-Server Operations**
  - Operations clients can send in requests to servers
- **Server-to-Client Operations**
  - Operations servers can send in requests to clients
- **Message Contents and Message Formats**
  - Request and Response protocol messages
- **Message Encoding**
  - Binary Tag-Type-Length-Value
- **Authentication**
  - See Profiles (Client Certificates)
- **Transport**
  - See Profiles (TLSv1.0 or TLSv1.2)
KMIP Fundamentals - Operations

- Establish
  - Create
  - Register
  - Create Key Pair
- Retrieve
  - Locate
  - Get Attribute
- Manage
  - Get Attribute List
  - Get Usage Allocation
  - Check
  - Obtain Lease
- State
  - Activate
  - Archive
  - Recover
  - Revoke
  - Destroy
- Info
  - Add Attribute
  - Modify Attribute
  - Delete Attribute
- Rotate
  - Re-Key
  - Re-Certify
- Server
  - Query
  - Poll
  - Cancel
- Client
  - Notify
  - Put
- Other
  - Discover Versions¹.¹
  - Validate
- Cryptographic
  - Decrypt¹.²
  - Sign¹.²
  - MAC¹.²
  - Hash¹.²
  - MACVerify¹.²
  - RNG Retrieve¹.²
  - RNG Seed¹.²
  - Encrypt¹.²
  - SignatureVerify¹.²
KMIP Fundamentals

- Managed Objects have a “Value”
  - Value is set at object creation
  - Value cannot be changed
  - Value may be “incomplete”
  - Value may be in varying formats
KMIP Fundamentals

- Managed Objects have an “Object Type”
  - Certificate
  - Symmetric Key
  - Public Key
  - Private Key
  - Split Key
  - Template *(Deprecated in KMIP 1.2)*
  - Secret Data
  - Opaque Object
  - PGP Key

KMIP Fundamentals

- Managed Objects have a set of “Attributes”
  - Every attribute has a string name
  - Every attribute has a type
  - May be simple types or complex types
  - Some set by server once and cannot be changed
  - Some set by client once and cannot be changed
  - Most are singleton (only one instance)
  - Server defined non-standard extensions are prefixed with “y-” in their string name
  - Client defined non-standard extensions are prefixed with “x-” in their string name
KMIP Fundamentals – Message Encoding

- Binary Tag-Type-Length-Value format
- Optional JSON and XML encoding in KMIP\textsuperscript{1.2}

\begin{itemize}
  \item Tag
  \item Type
  \item Length
\end{itemize}

\begin{itemize}
  \item Value
\end{itemize}

Cryptographic Usage Mask = Encrypt | Decrypt
### KMIP Fundamentals - TTLV

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>142 00 78 01 00 00 01 20 242 00 77 01 00 00 00 38</td>
</tr>
<tr>
<td>00000100</td>
<td>342 00 69 01 00 00 00 20 442 00 6a 02 00 00 00 04</td>
</tr>
<tr>
<td>00000200</td>
<td>00 00 00 01 00 00 00 00 542 00 6b 02 00 00 00 04</td>
</tr>
<tr>
<td>00000300</td>
<td>00 00 00 00 00 00 00 00 642 00 0d 02 00 00 00 04</td>
</tr>
<tr>
<td>00000400</td>
<td>00 00 00 01 00 00 00 00 742 00 0f 01 00 00 00 d8</td>
</tr>
<tr>
<td>00000500</td>
<td>842 00 5c 05 00 00 00 04 00 00 00 01 00 00 00 00</td>
</tr>
<tr>
<td>00000600</td>
<td>942 00 79 01 00 00 00 0c A42 00 57 05 00 00 00 04</td>
</tr>
<tr>
<td>00000700</td>
<td>00 00 00 02 00 00 00 00 B42 00 91 01 00 00 00 a8</td>
</tr>
<tr>
<td>00000800</td>
<td>C42 00 08 01 00 00 00 30 D42 00 0a 07 00 00 00 17</td>
</tr>
<tr>
<td>00000900</td>
<td>43 72 79 70 74 6f 67 72 61 70 68 69 63 20 41 6c</td>
</tr>
<tr>
<td>00000a00</td>
<td>67 6f 72 69 74 68 6d 00 E42 00 0b 05 00 00 00 04</td>
</tr>
<tr>
<td>00000b00</td>
<td>00 00 00 03 00 00 00 00 F42 00 08 01 00 00 00 30</td>
</tr>
<tr>
<td>00000c00</td>
<td>G42 00 0a 07 00 00 00 14 43 72 79 70 74 6f 67 72</td>
</tr>
<tr>
<td>00000d00</td>
<td>61 70 68 69 63 20 4c 65 6e 67 74 68 00 00 00 00</td>
</tr>
<tr>
<td>00000e00</td>
<td>H42 00 0b 02 00 00 00 04 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>00000f00</td>
<td>I42 00 08 01 00 00 00 30 J42 00 0a 07 00 00 00 18</td>
</tr>
<tr>
<td>00001000</td>
<td>43 72 79 70 74 6f 67 72 61 70 68 69 63 20 55 73</td>
</tr>
<tr>
<td>00001100</td>
<td>61 67 65 20 4d 61 73 6b K42 00 0b 02 00 00 00 04</td>
</tr>
<tr>
<td>00001200</td>
<td>00 00 00 0c 00 00 00 00</td>
</tr>
</tbody>
</table>
KMIP Fundamentals - XML

```xml
<RequestMessage>
  <RequestHeader>
    <ProtocolVersion>
      <ProtocolVersionMajor type="Integer" value="1"/>
      <ProtocolVersionMinor type="Integer" value="0"/>
    </ProtocolVersion>
    <BatchCount type="Integer" value="1"/>
  </RequestHeader>
  <BatchItem>
    <Operation type="Enumeration" value="Create"/>
    <RequestPayload>
      <ObjectType type="Enumeration" value="SymmetricKey"/>
      <TemplateAttribute>
        <Attribute>
          <AttributeName type="TextString" value="Cryptographic Algorithm"/>
          <AttributeValue type="Enumeration" value="AES"/>
        </Attribute>
        <Attribute>
          <AttributeName type="TextString" value="Cryptographic Length"/>
          <AttributeValue type="Integer" value="128"/>
        </Attribute>
        <Attribute>
          <AttributeName type="TextString" value="Cryptographic Usage Mask"/>
          <AttributeValue type="Integer" value="Decrypt Encrypt"/>
        </Attribute>
      </TemplateAttribute>
    </RequestPayload>
  </BatchItem>
</RequestMessage>
```
{"tag":"RequestMessage", "value":{
  "tag":"RequestHeader", "value":{
    "tag":"ProtocolVersion", "value":{
      "tag":"ProtocolVersionMajor", "type":"Integer", "value":"0x00000001"},
      "tag":"ProtocolVersionMinor", "type":"Integer", "value":"0x00000000"
    },
    "tag":"BatchCount", "type":"Integer", "value":"0x00000001"
  },
  "tag":"BatchItem", "value":{
    "tag":"Operation", "type":"Enumeration", "value":"Create"},
    "tag":"RequestPayload", "value":{
      "tag":"ObjectType", "type":"Enumeration", "value":"SymmetricKey"},
      "tag":"TemplateAttribute", "value":{
        "tag":"Attribute", "value":{
          "tag":"AttributeName", "type":"TextString", "value":"Cryptographic Algorithm"},
          "tag":"AttributeValue", "type":"Enumeration", "value":"AES"
        },
        "tag":"Attribute", "value":{
          "tag":"AttributeName", "type":"TextString", "value":"Cryptographic Length"},
          "tag":"AttributeValue", "type":"Integer", "value":"0x00000080"
        },
        "tag":"Attribute", "value":{
          "tag":"AttributeName", "type":"TextString", "value":"Cryptographic Usage Mask"},
          "tag":"AttributeValue", "type":"Integer", "value":"Decrypt|Encrypt"
        }
      }
    }
  }
}
OASIS KMIP vendor

IMPLEMENTATION ERRORS
Implementation Errors

- Simple
  - Invalid Padding
  - Invalid Encoding
  - Invalid Tag Values
  - Invalid Field Order
  - Invalid TLS usage
  - Missing Mandatory
  - Mandating Optional
  - Invalid sign
Implementation Errors

- Complex
  - Core concepts omitted
  - Special interpretation added
  - Conceptual confusion (Templates)
  - Unusual feature set selection
  - Assumed message sequences and content
Implementation Errors

*Simple invalid encoding errors*

- The specification includes clear text on encoding
- The specification includes examples of each encoding
- The KMIP 1.0 Test Cases include the hexadecimal request and response sequences
- Almost every vendor gets one or more of the encoding items wrong
Implementation Errors

9.1.1.3 Item Length

An Item Length is a 32-bit binary integer, transmitted big-endian, containing the number of bytes in the Item Value.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>Varies, multiple of 8</td>
</tr>
<tr>
<td>Integer</td>
<td>4</td>
</tr>
<tr>
<td>Long Integer</td>
<td>8</td>
</tr>
<tr>
<td>Big Integer</td>
<td>Varies, multiple of 8</td>
</tr>
<tr>
<td>Enumeration</td>
<td>4</td>
</tr>
<tr>
<td>Boolean</td>
<td>8</td>
</tr>
<tr>
<td>Text String</td>
<td>Varies</td>
</tr>
<tr>
<td>Byte String</td>
<td>Varies</td>
</tr>
<tr>
<td>Date-Time</td>
<td>8</td>
</tr>
<tr>
<td>Interval</td>
<td>4</td>
</tr>
</tbody>
</table>

Actual Implementation Errors

- No padding
- Padding before rather than at end of value
- Padding missing for some types
- Padding added for types that do not require padding

If the Item Type is Structure, then the Item Length is the total length of all of the sub-items contained in the structure, including any padding. If the Item Type is Integer, Enumeration, Text String, Byte String, or Strings SHALL be padded with the minimal number of bytes following the Item Value to obtain a multiple Value.
Implementation Errors - Solution

*Simple invalid encoding*

- Accept that adding more specification text does not fix this issue
- Accept that adding more examples of encoding are the same as adding more specification text – they are simply either not read or not read carefully
- Accept that test cases seem to be ignored more often than they are used
Implementation Errors - Solution

Simple invalid encoding errors

- Test interoperability between implementations
  - More plug-fests
  - More interop-events
  - More tests defined in more approachable manner
- Formal conformance testing program
  i.e. more events and wider scope
Implementation Errors

Special interpretation or conceptual confusion

- Adding semantics that don’t exist – leaping beyond the spec to non-interoperable solutions
  - Using *Templates* for policy management
  - Automatically creating objects during search
  - Ignoring Password fields (accept anything)
  - Requiring Names
  - Forcing restricted set of characters in Names
Implementation Errors - Solution

Special interpretation or conceptual confusion

- Deprecated Templates as of KMIP 1.2
- Require explicit indication for create-when-searching if really necessary
- Adding Alternate Name and “vendor education”
- Expanding testing of Names which exceed arbitrary restrictions (spaces, punctuation, etc)
- More test cases and profiles
- Flexible interpretation in servers
Implementation Errors

Assumed message sequences and content

- Pattern matching rather than understanding
  - Ignoring most of the message content
  - Assuming fixed list of fields in fixed order for non-ordered lists
  - Assuming fixed sequence of request / response items
- Pre-canned responses with minimal substitution
- Ignoring protocol version information
Implementation Errors - Solution

Assumed message sequences and content

- Detect this sort of implementation
- Determine limitations of the approach
- Expand on testing to require more semantic processing rather than simple syntax
- More test cases and profiles
Guidance for key vendor issues in

KEY MANAGEMENT
Guidance

- Fundamental Requirements
  - Don’t lose the keys
    - Don’t break the device or application using keys
  - Don’t stop serving keys when they are needed
    - Don’t stop the device or application keys from working
  - Don’t give the keys to the wrong person
    - Don’t break the purpose of adding encryption by undoing the security properties
Guidance

- **Context**
  - Context free key management is low value
  - Anonymous keys don’t allow for active security management or meaningful auditing
  - How much context can be provided
    - KMIP has no fundamental (practical) limits on attaching context and cross-relating keys
Guidance

- Clear requirements
  - What do you want for interoperability now
  - What are you likely to want in the future
  - How do your products use key management
  - How will your security administrators use key management
  - What are your target number of keys and access patterns
    - Performance radically varies between vendors
Danger signs in vendor approaches to KEY MANAGEMENT
Danger Signs

- Only indication of KMIP support is in product data sheet
- Vendor-specific implementation and no interoperability indicators (no plug-fest, no-interop, no conformance report, no vendor-to-vendor KMIP integration claims)
Danger Signs

- Key management integrations listed without making it clear **which protocol** is being used
  - Claims of legacy protocol integrations not separated from KMIP integrations
  - Server supports KMIP; Client supports server does not mean client uses KMIP
- Capabilities not clearly separated between vendor protocol and KMIP
  - Creative marketing messages
The importance of vendor-independent

CONFORMANCE TESTING
KMIP Conformance Testing - Intent

- The SNIA – Storage Security Industry Forum (SSIF) launched the program in response to market demand
- The program enables organizations to shortlist vendor KMIP solutions based on support for specific usage scenarios and interoperability
- Enables organizations to verify vendor claims
- Value provided by a truly independent test team
KMIP Conformance Testing - Profiles

- The KMIP Technical Committee defines Profiles
- Normative documents specify minimum set of supported functionality
- Contain expected requests and responses
- Cover a range of deployment scenarios

**KMIP Profiles**

- Advanced Cryptographic Client & Server
- Advanced Symmetric Key Foundry Client & Server
- Asymmetric Key Lifecycle Client & Server
- Baseline Client & Server Basic
- Baseline Client & Server TLSv1_2
- Basic Cryptographic Client & Server
- Basic Symmetric Key Foundry Client & Server
- HTTPS, JSON, XML Client & Server
- Intermediate Symmetric Key Foundry Client & Server
- Opaque Managed Object Store Client & Server
- RNG Cryptographic Client & Server
- Storage Array With SED Client & Server
- Suite-B MinLOS_128 Client & Server
- Suite-B MinLOS_192 Client & Server
- Symmetric Key Lifecycle Client & Server
- Tape Library Client & Server
- Complete Server
KMIP Conformance Testing - Method

- Implementations are made available to the test team
  - Implementations may be tested onsite or remotely
- Test team operates under the SSIF’s direction but testing information is kept completely confidential
- Test Report is provided to the customer
KMIP Conformance Testing - Client

Customer Storage Product (KMIP Client)  SSIF Test Infrastructure
KMIP Profile – Example (SASED)

```xml
# TIME 0
0001 <RequestMessage>
0002  <RequestHeader>
0003   <ProtocolVersion>
0004     <ProtocolVersionMajor type="Integer" value="1"/>
0005     <ProtocolVersionMinor type="Integer" value="0"/>
0006   </ProtocolVersion>
0007   <BatchCount type="Integer" value="1"/>
0008 </RequestHeader>
0009 </RequestItem>
0010 <Operation type="Enumeration" value="Query"/>
0011 <RequestPayload>
0012   <QueryFunction type="Enumeration" value="QueryServerInformation"/>
0013   </RequestPayload>
0014 </RequestItem>
0015 </RequestMessage>

0018 <ResponseMessage>
0019  <ResponseHeader>
0020   <ProtocolVersion>
0021     <ProtocolVersionMajor type="Integer" value="1"/>
0022     <ProtocolVersionMinor type="Integer" value="0"/>
0023   </ProtocolVersion>
0024   <TimeStamp type="DateTime" value="2013-04-25T16:53:03+00:00"/>
0025   <BatchCount type="Integer" value="1"/>
0026 </ResponseHeader>
0027 <ResponseItem>
0028 <Operation type="Enumeration" value="Query"/>
0029 <ResultStatus type="Enumeration" value="Success"/>
0030 <ResponsePayload>
0031   <Operation type="Enumeration" value="Query"/>
0032   <Operation type="Enumeration" value="Locate"/>
0033   <Operation type="Enumeration" value="Destroy"/>
0034   <Operation type="Enumeration" value="Get"/>
0035   <Operation type="Enumeration" value="Register"/>
0036   <Operation type="Enumeration" value="GetAttributes"/>
0037   <Operation type="Enumeration" value="GetAttributeList"/>
0038   <Operation type="Enumeration" value="AddAttribute"/>
0039   <ObjectType type="Enumeration" value="SecretData"/>
0040   <ObjectType type="Enumeration" value="Template"/>
0041   <VendorIdentification type="TextString" value="server-vendor.com"/>
0042   <ServerInformation>
0043 </ServerInformation>
0044 </ResponsePayload>
0045 </ResponseItem>
0046 </ResponseMessage>
```
KMIP Conformance Testing - Server

Customer Storage Server (KMIP Server) → SSIF Test Infrastructure

[Diagram showing the relationship between the Customer Storage Server (KMIP Server) and the SSIF Test Infrastructure]
KMIP Conformance Testing - Results

- Results remain entirely confidential to customer and conformance test team until results are published.
- Test results are published with customer’s permission by SNIA SSIF.
- Only the successful results (supported profiles) appear on the results page.
  - Failures and/or non-supported profiles are not stated.
## KMIP Conformance Testing Results

Note: This SNIA webpage lists vendor products that have successfully passed one or more KMIP Conformance Tests which are performed by SNIA against KMIP profiles and associated test cases as published by the OASIS KMIP Technical Committee. SNIA is not liable for any damages or claims related to use of any vendor products.

<table>
<thead>
<tr>
<th>COMPANY</th>
<th>PRODUCT</th>
<th>PROFILE(s) TESTED</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hewlett Packard Enterprise</td>
<td>HP StoreEver MSL2024 Version 6.41</td>
<td>Tape Library V1.2</td>
<td>12-Apr-2016</td>
</tr>
<tr>
<td>Hewlett Packard Enterprise</td>
<td>HP StoreEver MSL4040 Version 8.91</td>
<td>Tape Library V1.2</td>
<td>12-Apr-2016</td>
</tr>
</tbody>
</table>

Snapshot taken from: [http://www.snia.org/forums/SSIF/kmip/results](http://www.snia.org/forums/SSIF/kmip/results)
Summary on multi-vendor

KEY MANAGEMENT
Summary

- Capability and claims vary substantially
- Verify claims – don’t make assumptions
- Interoperability is only actually achieved when products work together
- Conformance testing programs provide assurance and reduce the burden of point-to-point testing
Multi-Vendor Key Management with KMIP

Tim Hudson
CTO & Technical Director

CRYPTOFT

tjh@cryptsoft.com