Key Management Interoperability Protocol (KMIP)

April 2nd, 2009

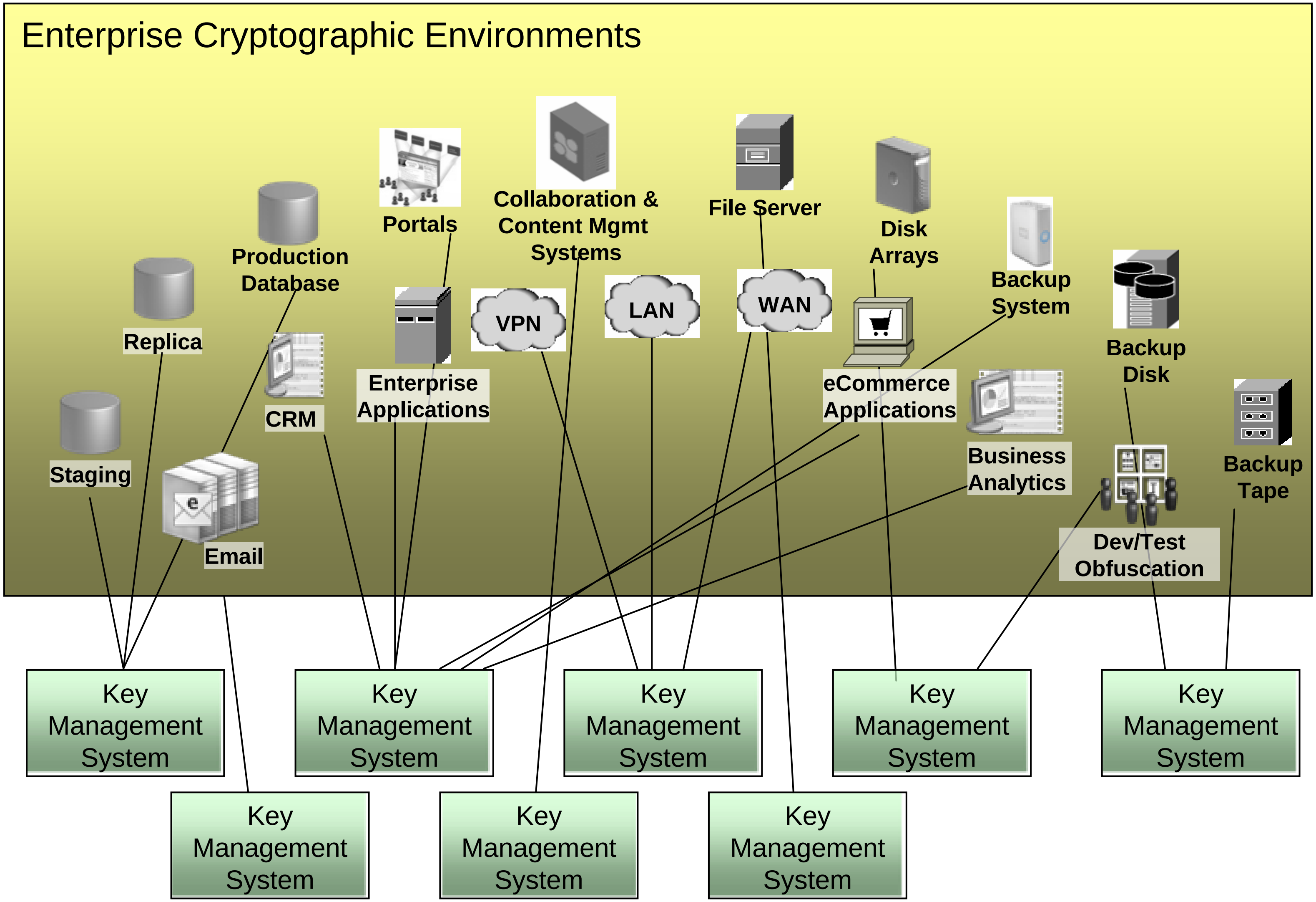
Agenda

- The Need for Interoperable Key Management
- KMIP Overview
- KMIP Specification
- KMIP Use Cases

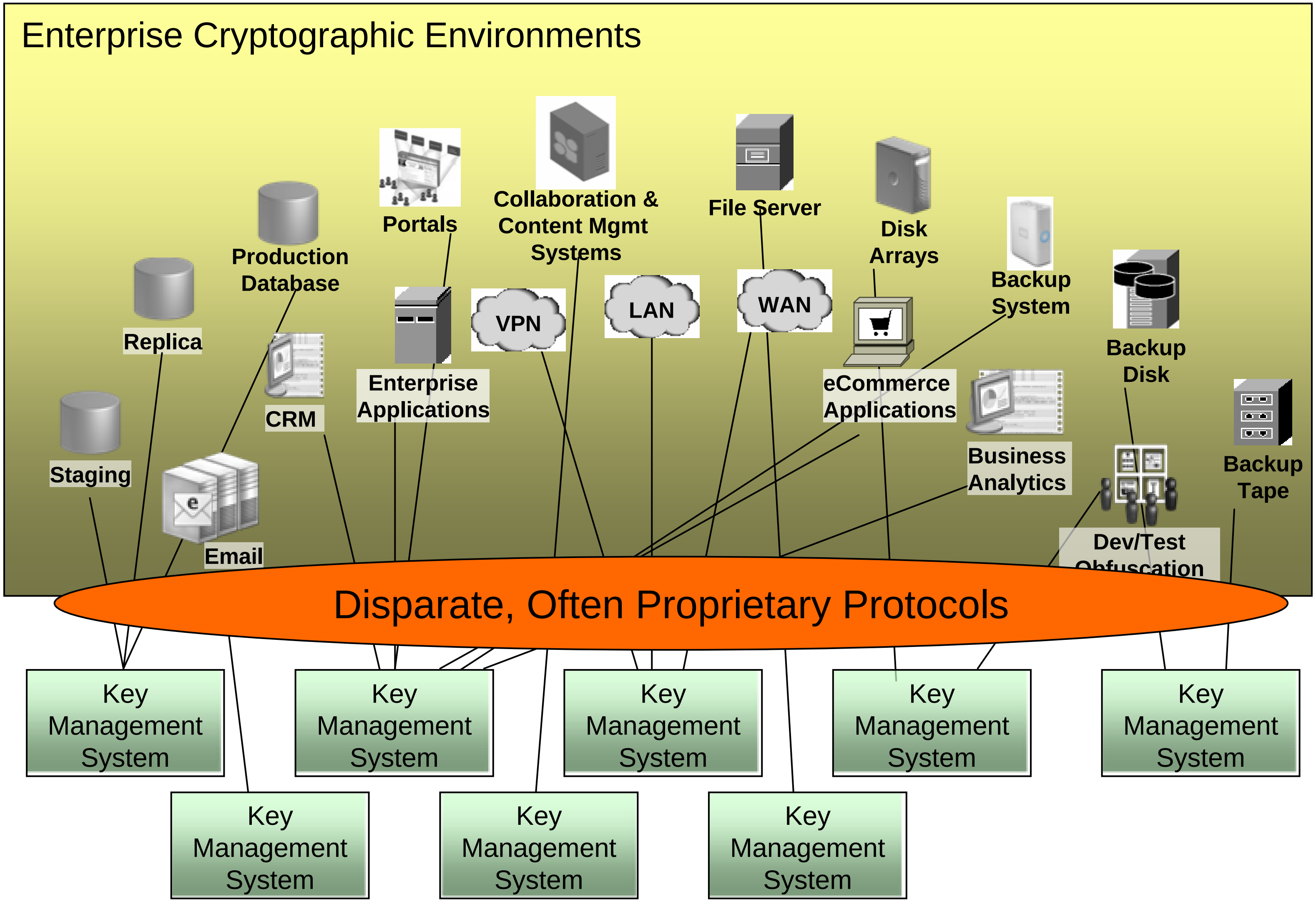
The Need for Interoperable Key Management

- Today's enterprises operate in increasingly complex, multi-vendor environments.
- Enterprises need to deploy better encryption across the enterprise.
- A key hurdle in IT managers deploying encryption is their ability to recover the encrypted data.
- Today, many companies deploy separate encryption systems for different business uses – laptops, storage, databases and applications – resulting in:
  - Cumbersome, often manual efforts to manage encryption keys
  - Increased costs for IT
  - Challenges meeting audit and compliance requirements
  - Lost data

Often, Each Cryptographic Environment Has Its Own Key Management System



Often, Each Cryptographic Environment Has Its Own Protocol



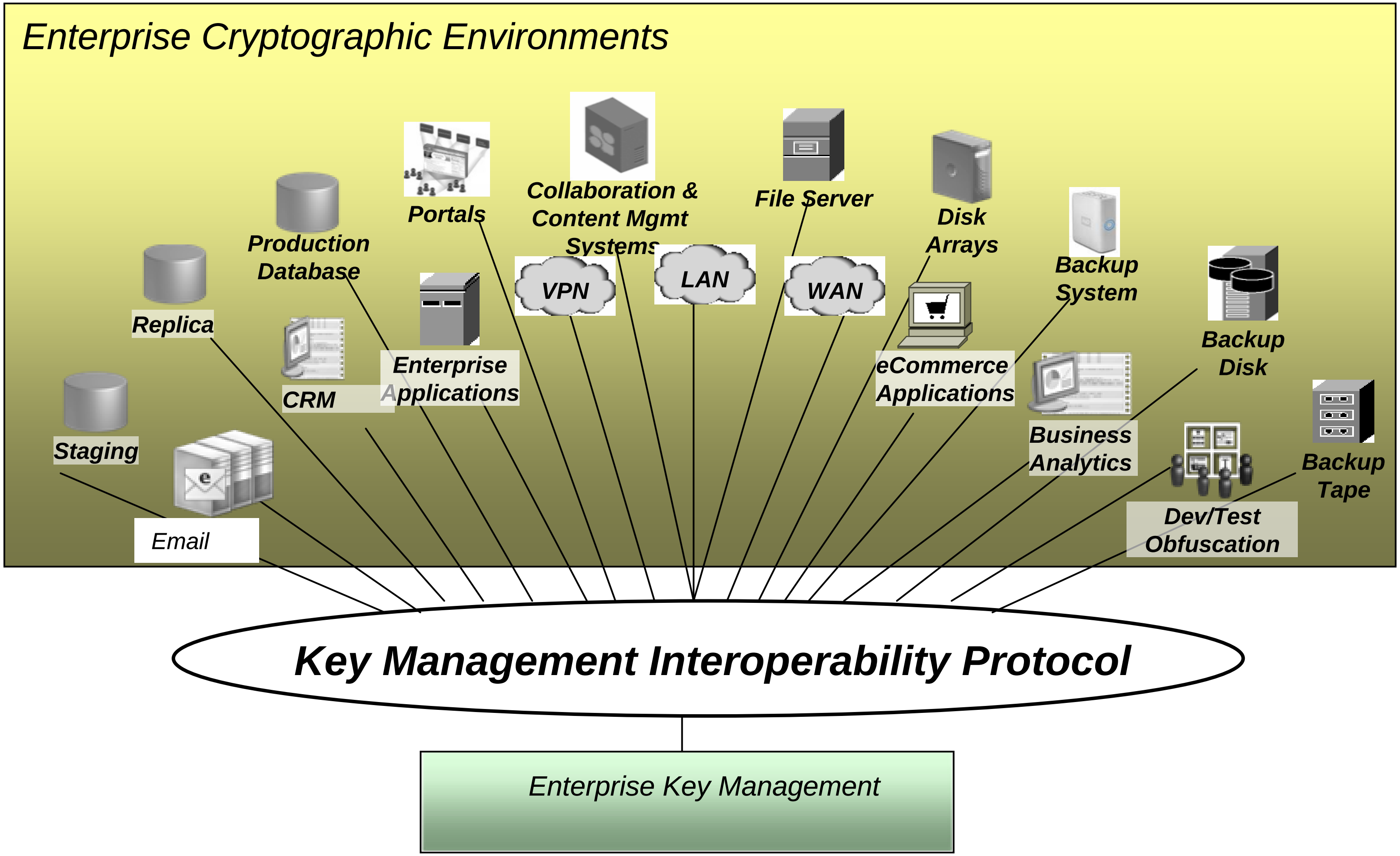
KMIP Overview

What is KMIP

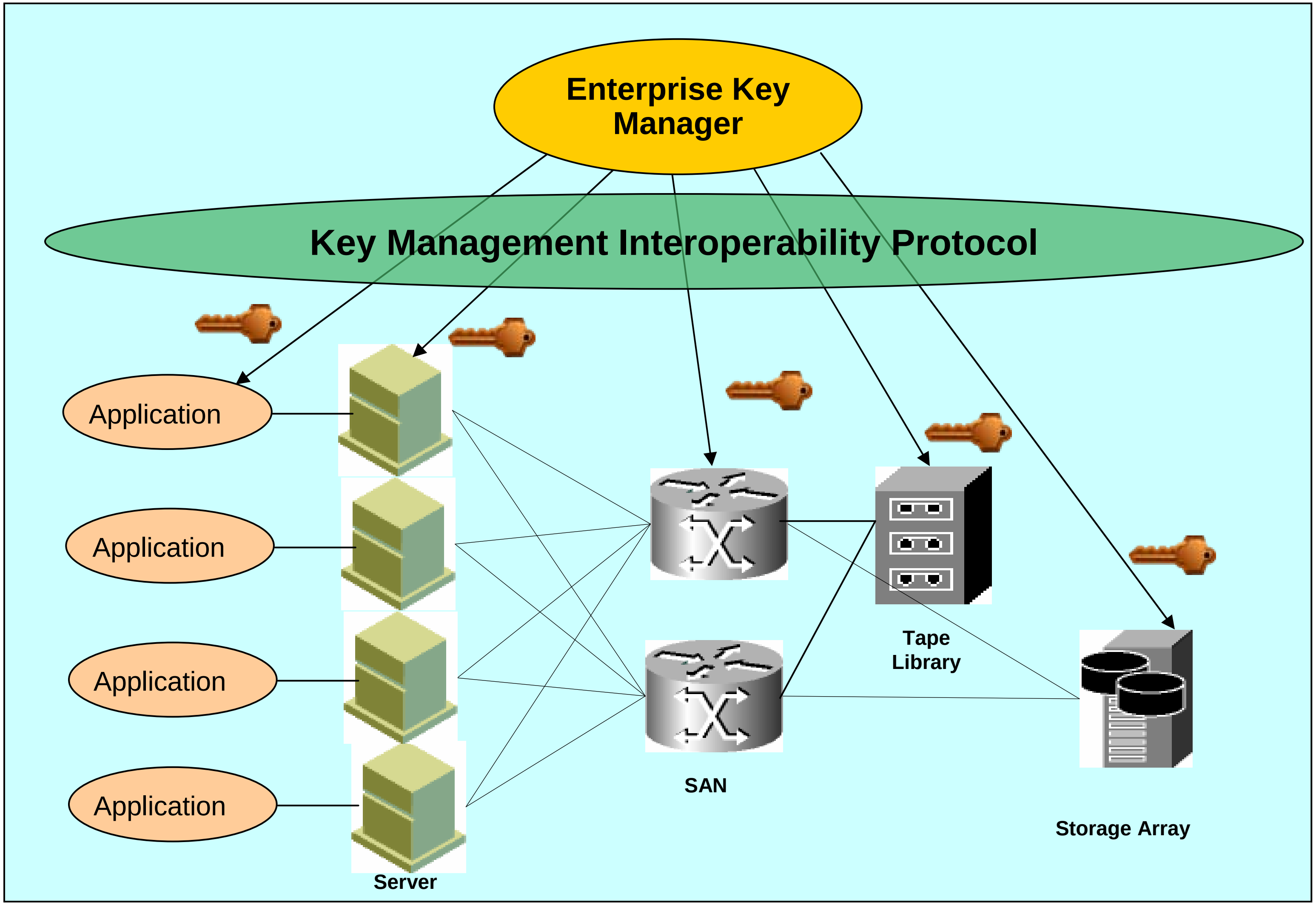
- The Key Management Interoperability Protocol (KMIP) enables key lifecycle management. KMIP supports legacy and new encryption applications, supporting symmetric keys, asymmetric keys, digital certificates, and other "shared secrets." KMIP offers developers templates to simplify the development and use of KMIP-enabled applications.

- KMIP defines the protocol for encryption client and key-management server communication. Key lifecycle operations supported include generation, submission, retrieval, and deletion of cryptographic keys. Vendors will deliver KMIP-enabled encryption applications that support communication with compatible KMIP key-management servers.

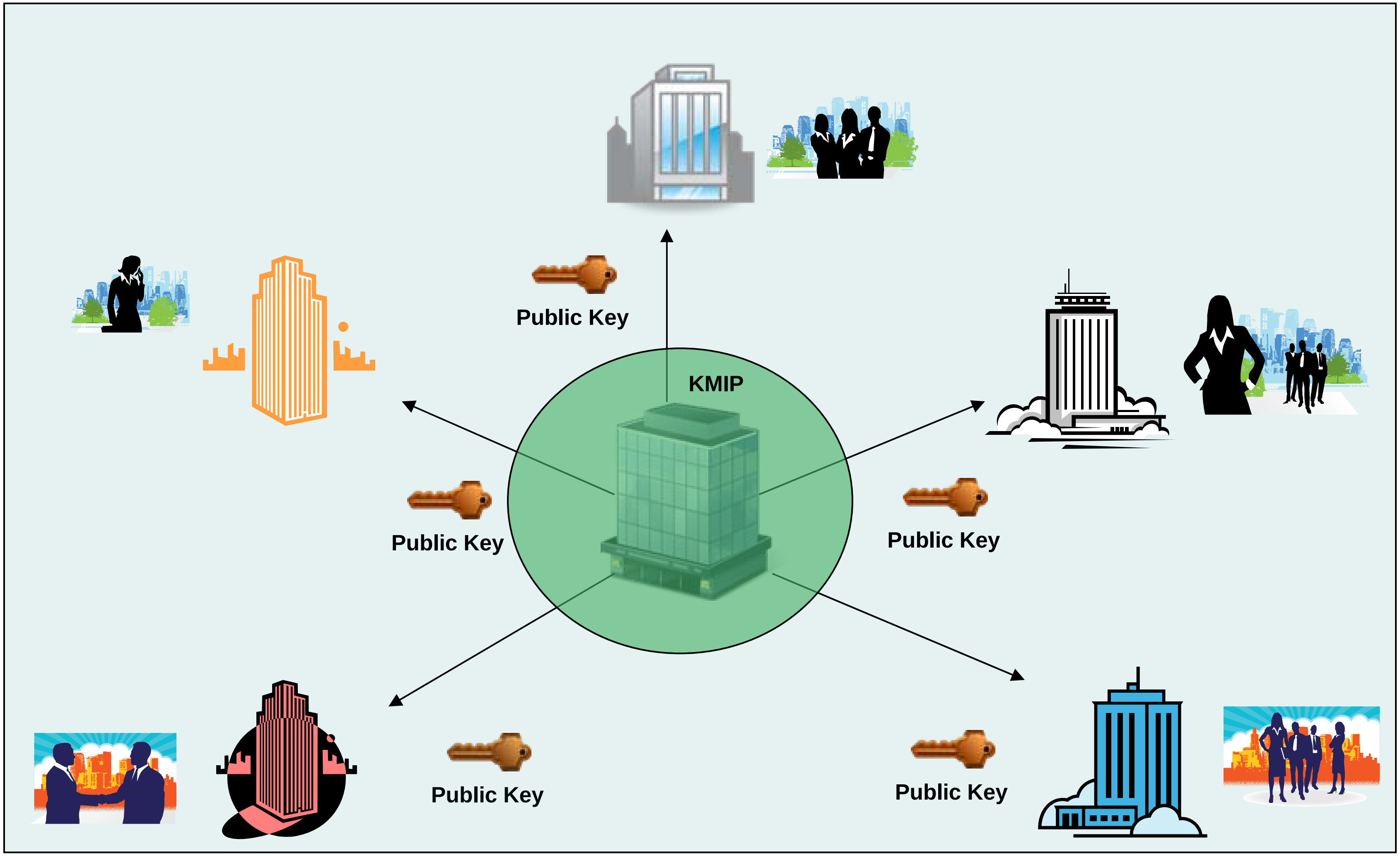
KMIP: Single Protocol Supporting Enterprise Cryptographic Environments



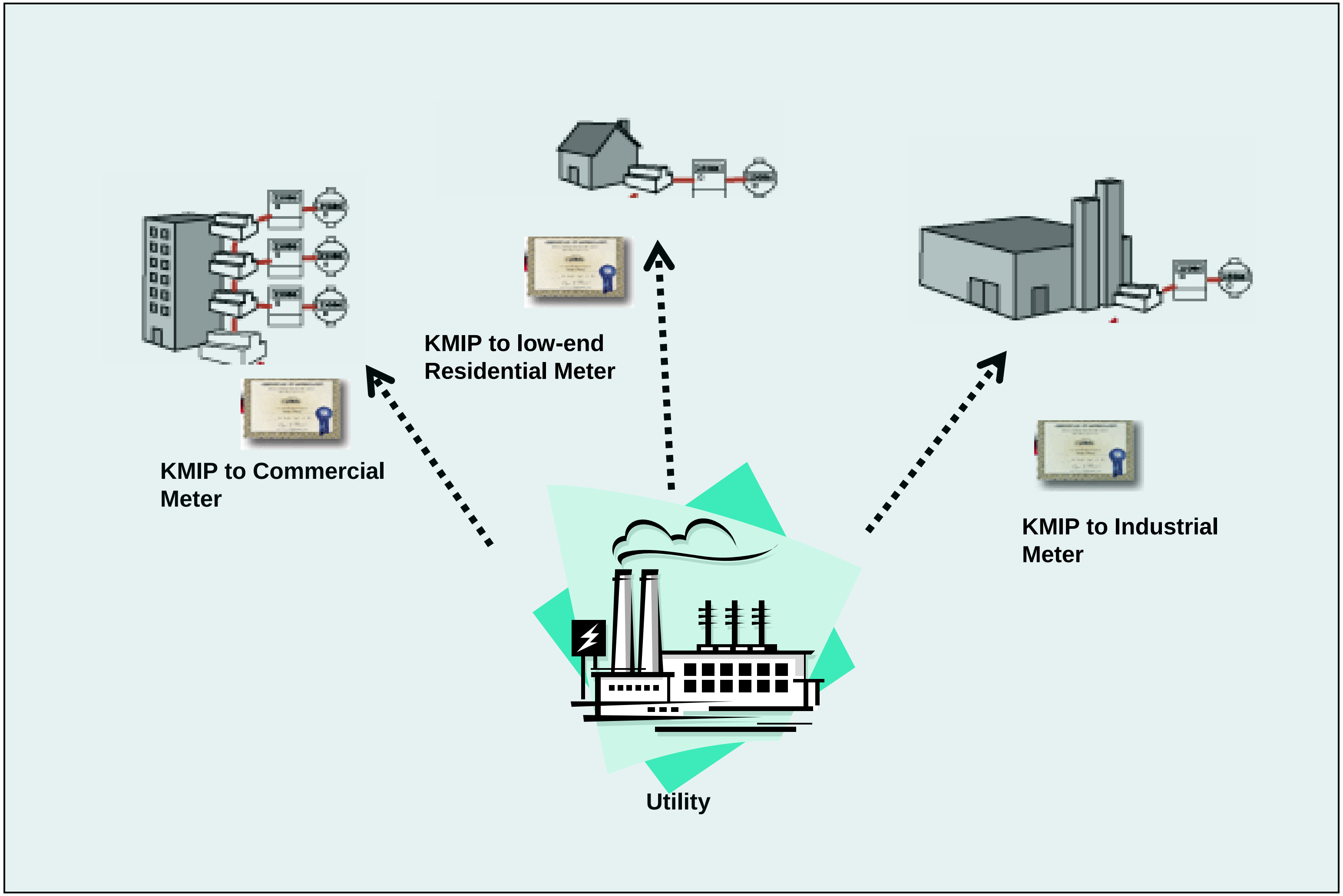
KMIP: Symmetric Encryption Keys



KMIP: Asymmetric Keys



KMIP: Digital Certificates



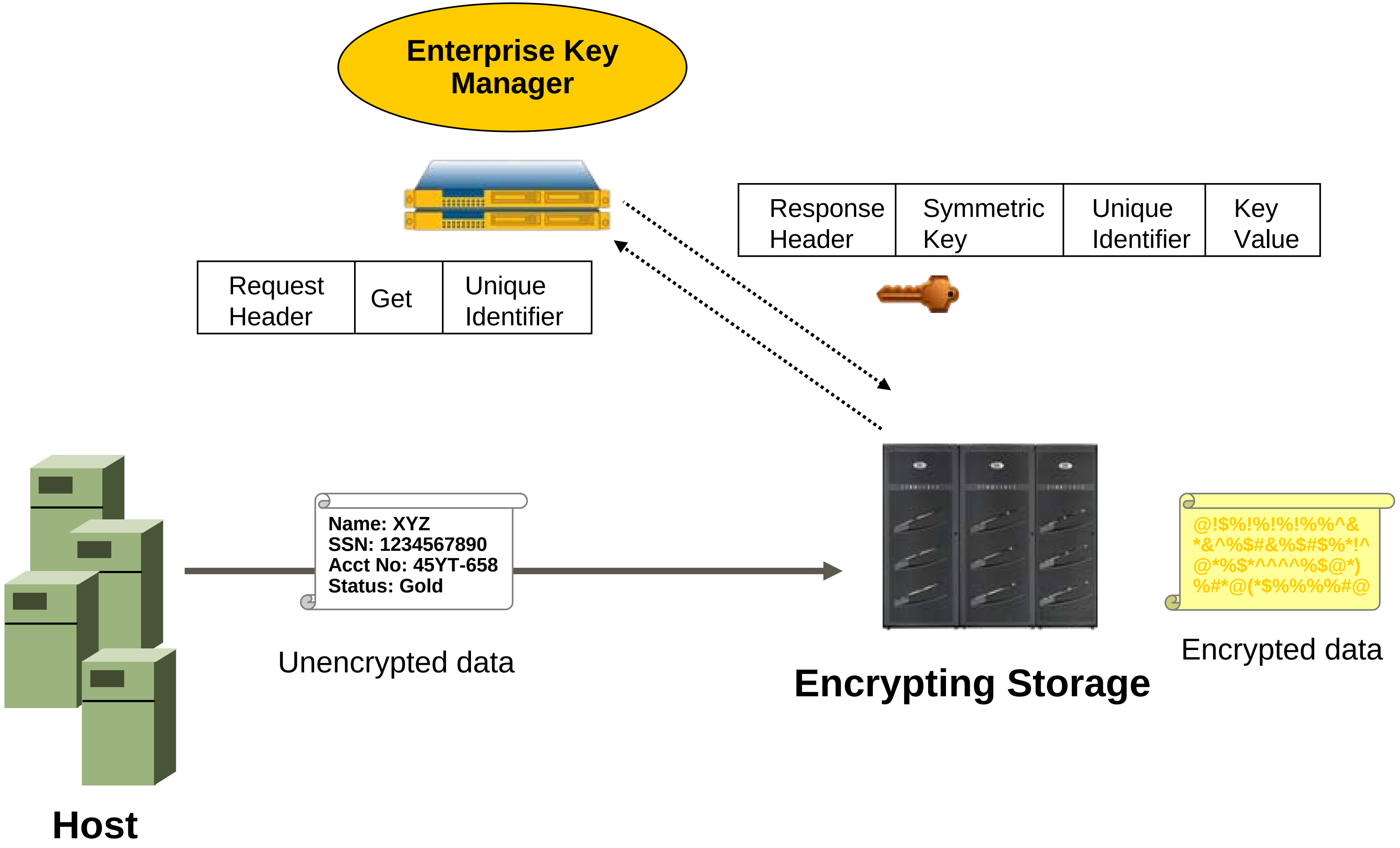
KMIP to low-end Residential Meter

KMIP to Commercial Meter

KMIP to Industrial Meter

Utility

KMIP Request / Response Model



Response Header	Symmetric Key	Unique Identifier	Key Value

Request Header	Get	Unique Identifier

Name: XYZ
SSN: 1234567890
Acct No: 45YT-658
Status: Gold

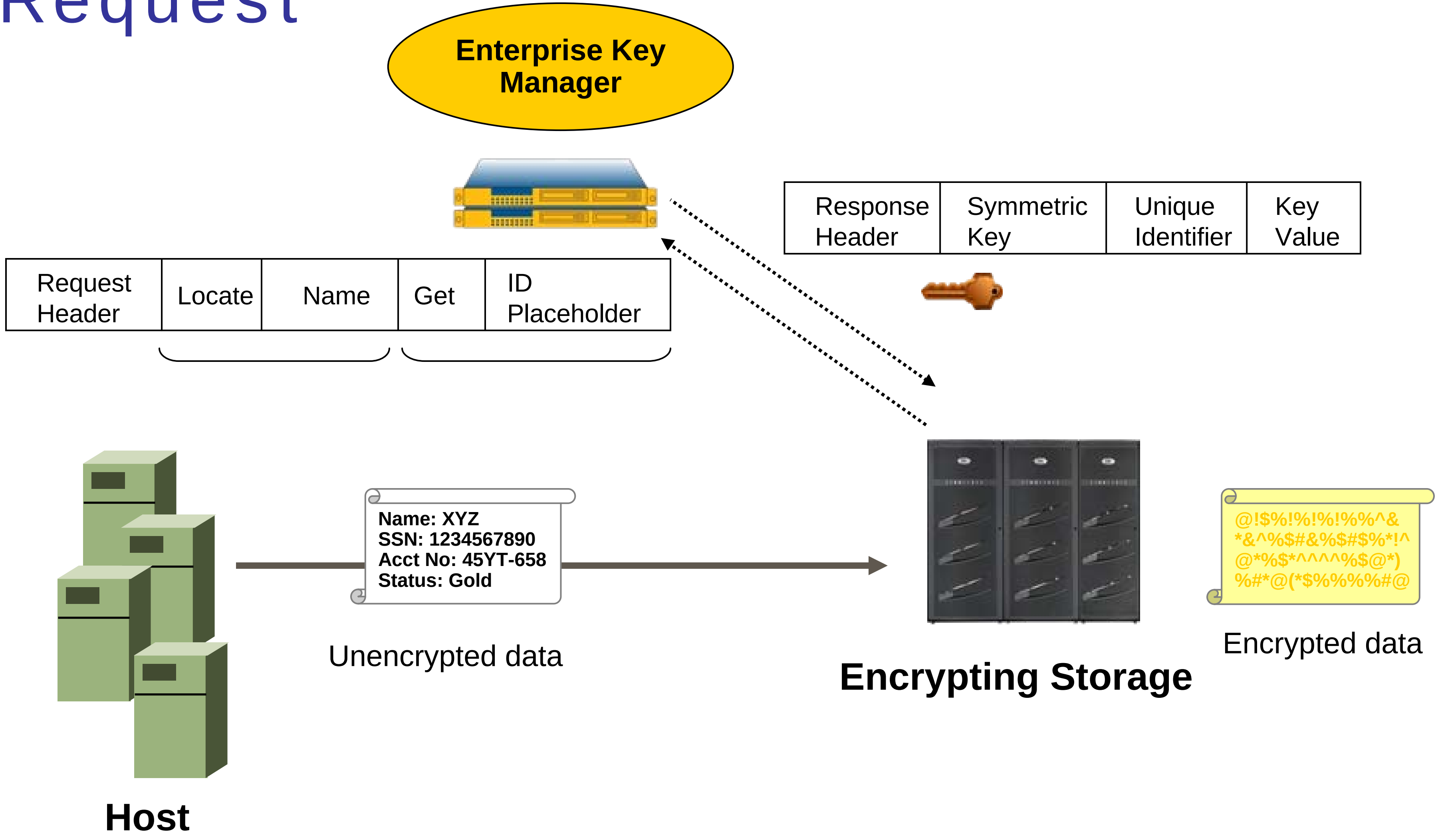
Unencrypted data

Encrypted data

Encrypting Storage

Host

Supporting Multiple Operations per Request



Enterprise Key Manager

Request Header	Locate	Name	Get	ID Placeholder

Response Header	Symmetric Key	Unique Identifier	Key Value

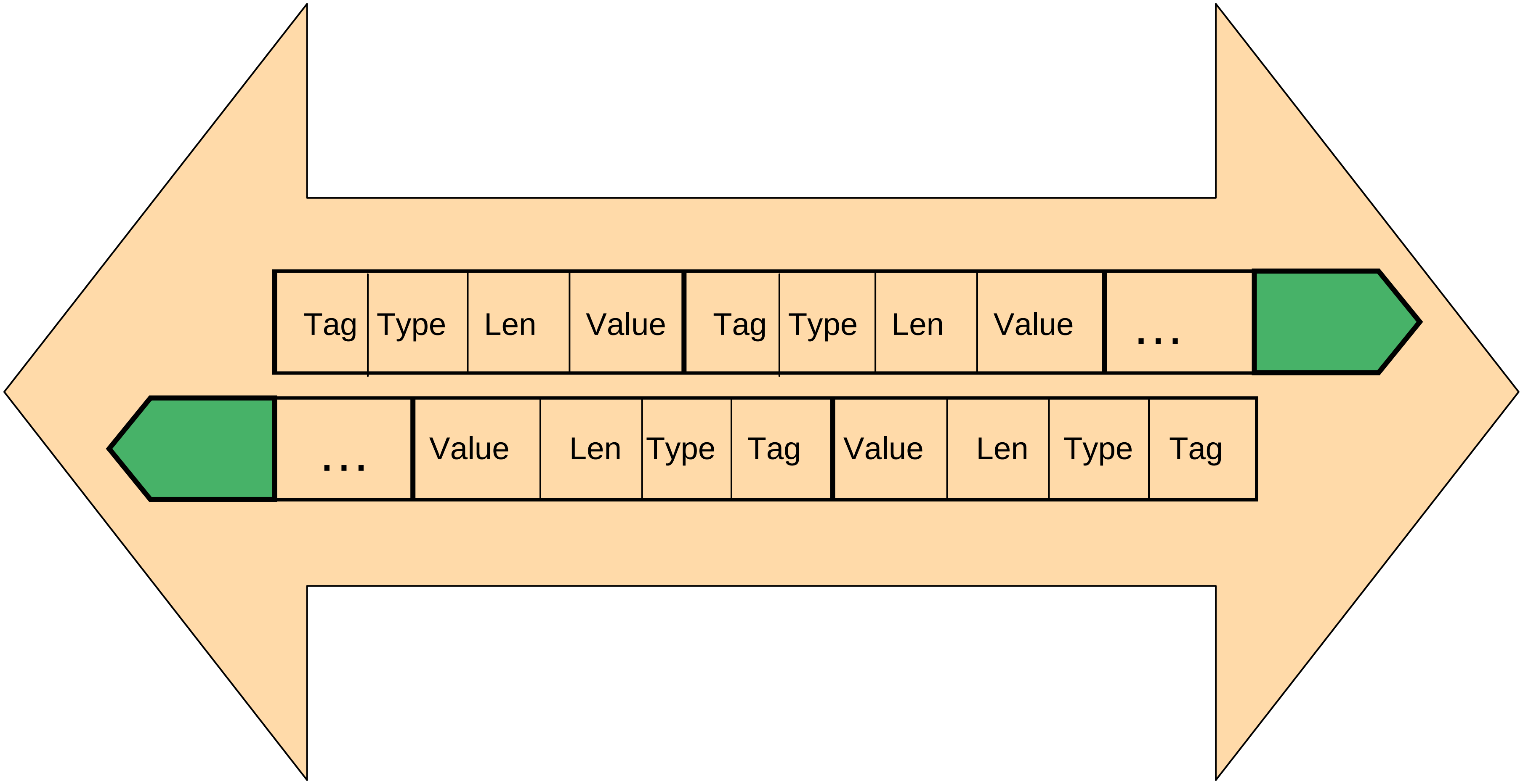
Name: XYZ
SSN: 1234567890
Acct No: 45YT-658
Status: Gold

Unencrypted data

@!$%!%!%!%%^&
&^%$#&%$#$%!^
@*%$*^^^^%$@*)
%#*@(*$%%%%#@

Encrypted data

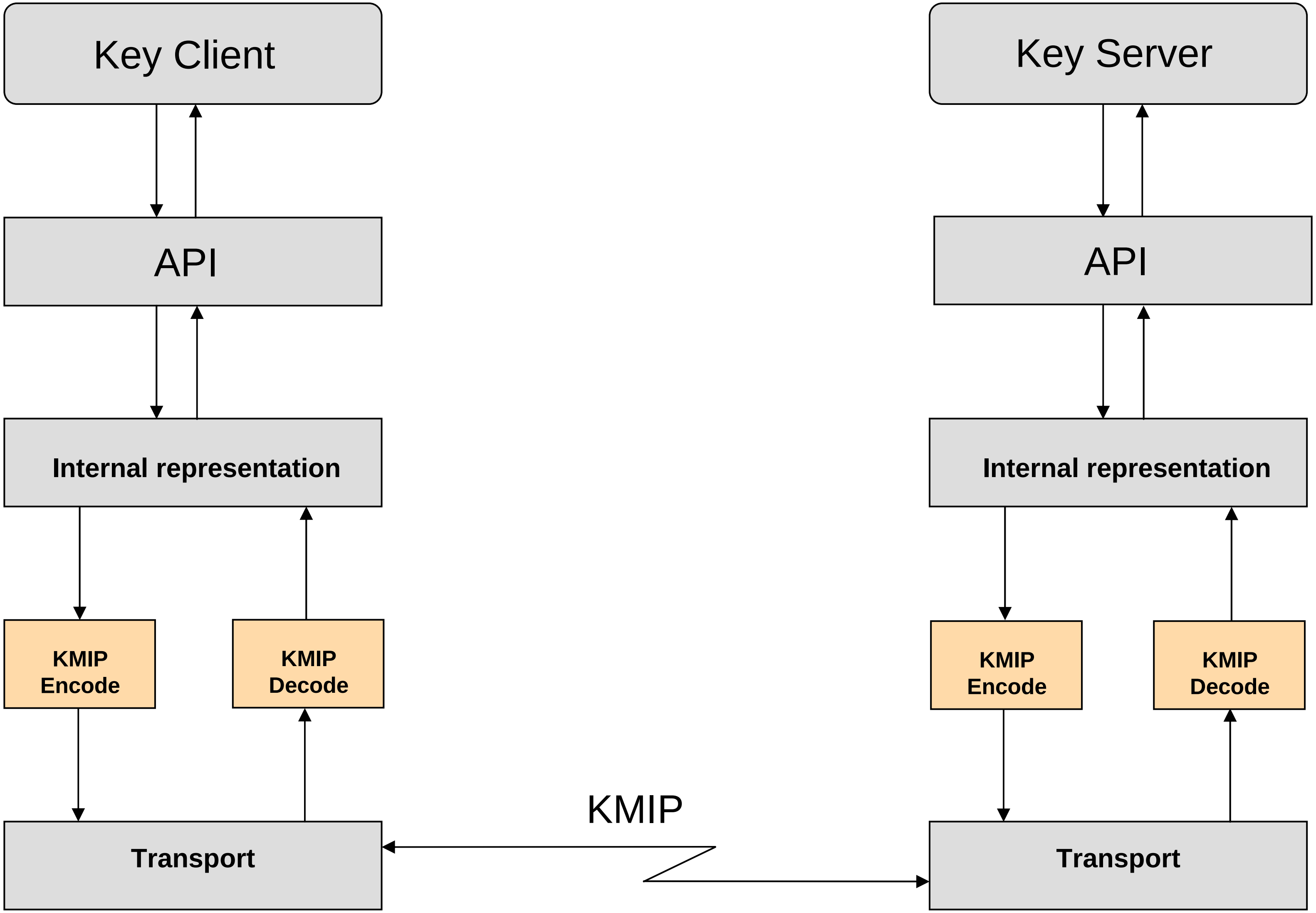
Messages in TTLV Format



Tag | Type | Len | Value | Tag | Type | Len | Value | …

… | Value | Len | Type | Tag | Value | Len | Type | Tag

Transport-Level Encoding



OASIS KMIP Technical Committee

- OASIS (Organization for the Advancement of Structured Information Standards) is a not-for-profit consortium that drives the development, convergence and adoption of open standards for the global information society.

- KMIP Technical Committee chartered in March 2009
  - “The KMIP TC will develop specification(s) for the interoperability of Enterprise Key Management (EKM) services with EKM clients. The specifications will address anticipated customer requirements for key lifecycle management (generation, refresh, distribution, tracking of use, life-cycle policies including states, archive, and destruction), key sharing, and long-term availability of cryptographic objects of all types (public/private keys and certificates, symmetric keys, and other forms of “shared secrets”) and related areas.”

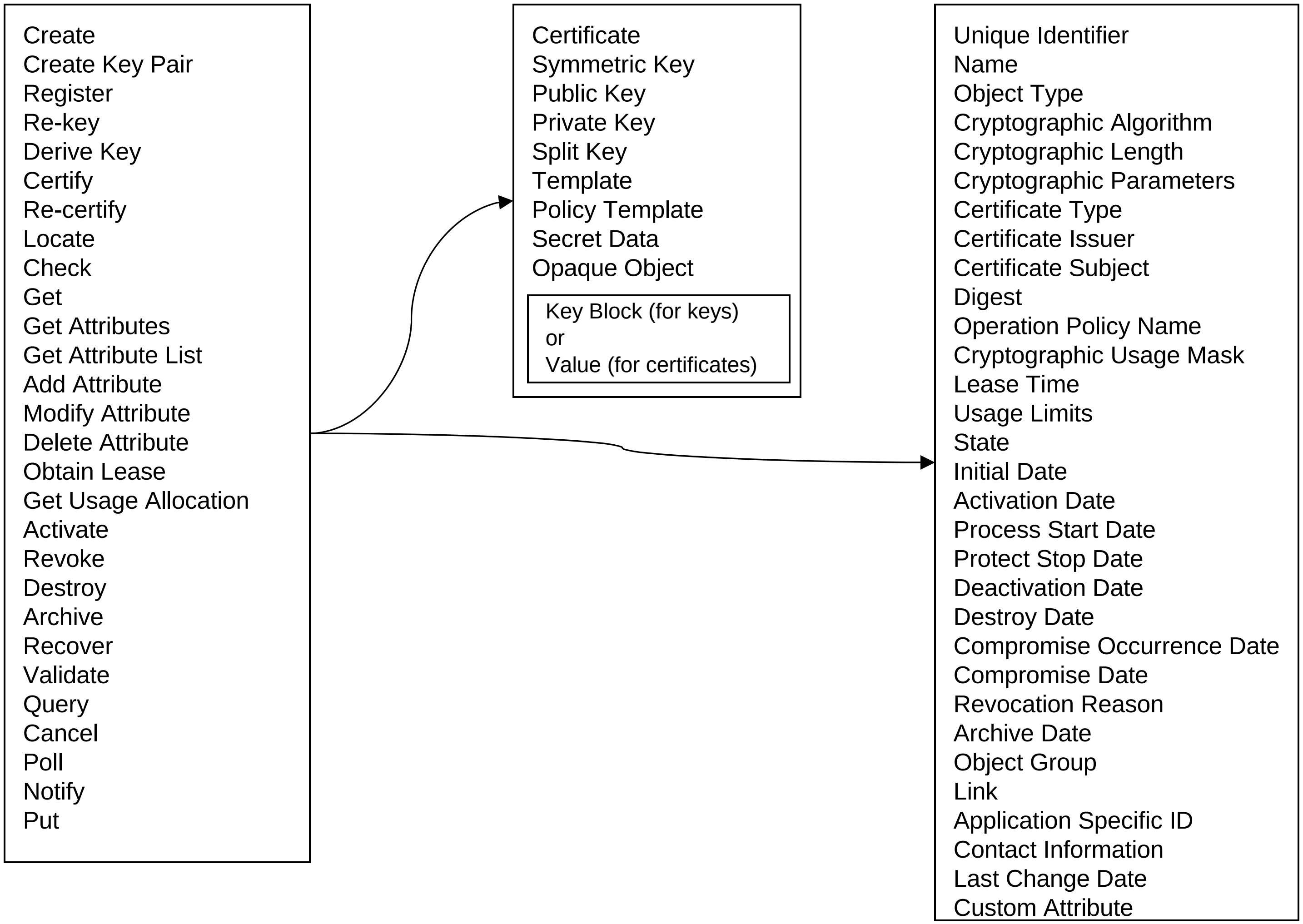
OASIS KMIP TC cont’d

- KMIP TC IPR mode is Royalty Free on RAND
- First TC meeting face-to-face April 24th, San Francisco CA, must sign up by April 9th to have voting rights.
- Reading material:
  - http://xml.coverpages.org/KMIP/KMIP-FAQ.pdf
  - http://xml.coverpages.org/KMIP/KMIP-UsageGuide-v0.98-final.pdf

KMIP Specification

http://xml.coverpages.org/KMIP/KMIP-v0.98-final.pdf

KMIP defines a set of Operations that apply to Managed Objects that consist of Attributes and possibly cryptographic material



Protocol Operations

Create
Create Key Pair
Register
Re-key
Derive Key
Certify
Re-certify
Locate
Check
Get
Get Attributes
Get Attribute List
Add Attribute
Modify Attribute
Delete Attribute
Obtain Lease
Get Usage Allocation
Activate
Revoke
Destroy
Archive
Recover
Validate
Query
Cancel
Poll
Notify
Put

Managed Objects

Certificate
Symmetric Key
Public Key
Private Key
Split Key
Template
Policy Template
Secret Data
Opaque Object

Key Block (for keys)
or
Value (for certificates)

Object Attributes

Unique Identifier
Name
Object Type
Cryptographic Algorithm
Cryptographic Length
Cryptographic Parameters
Certificate Type
Certificate Issuer
Certificate Subject
Digest
Operation Policy Name
Cryptographic Usage Mask
Lease Time
Usage Limits
State
Initial Date
Activation Date
Process Start Date
Protect Stop Date
Deactivation Date
Destroy Date
Compromise Occurrence Date
Compromise Date
Revocation Reason
Archive Date
Object Group
Link
Application Specific ID
Contact Information
Last Change Date
Custom Attribute

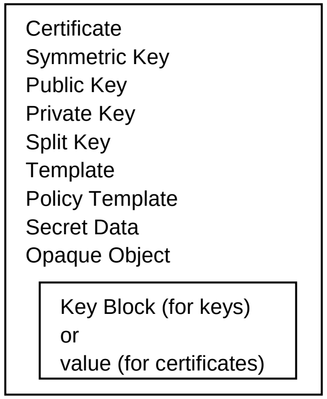
KMIP Base Objects

- Base Objects are:
  - Components of Managed Objects:
    - Attribute, identified by its Attribute Name
    - Key Block, containing the Key Value, either
      - in the clear, either in raw format, or as a transparent structure
      - or “wrapped” using Encrypt, MAC/Sign, or combinations thereof
      - possibly together with some attribute values
  - Elements of protocol messages:
    - Credential, used in protocol messages
  - Parameters of operations:
    - Template-Attribute, containing template names and/or attribute values, used in operations

KMIP Managed Objects

- Managed Cryptographic Objects
  - Certificate, with type and value
  - Symmetric Key, with Key Block
  - Public Key, with Key Block
  - Private Key, with Key Block
  - Split Key, with parts and Key Block
  - Secret Data, with type and Key Block

Managed Objects



Certificate
Symmetric Key
Public Key
Private Key
Split Key
Template
Policy Template
Secret Data
Opaque Object

Key Block (for keys)
or
value (for certificates)

- Managed Objects
  - Template and Policy Template:
    - Template has a subset of Attributes that indicate **what an object** created from such a template **is**
    - Policy Template has a subset of Attributes that indicate **how an object** created from such a template **can be used**
    - Note that (Policy) Templates have nothing except Attributes: for convenience these Attributes are included in the (Policy) Template structure too.
  - Opaque Object, without Key Block

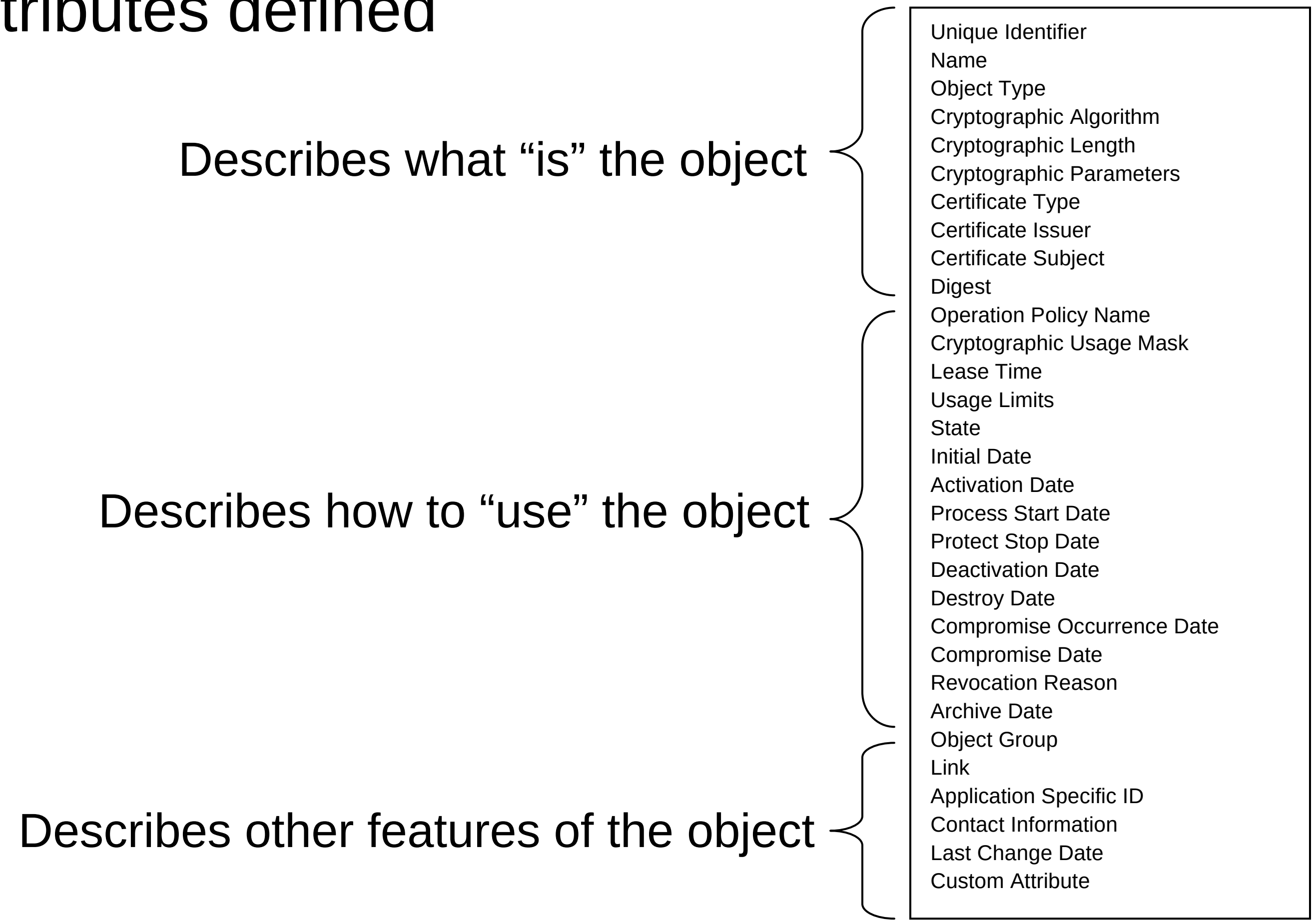
KMIP Attributes

- Attributes contain the “meta data” of a Managed Object
  - Its Unique Identifier, State, etc
  - Attributes can be searched with the Locate operation, as opposed to the content of the Managed Object

- Setting/modifying/deleting Attributes
  - Only some of the Attributes are set with specific values at object creation, depending on the object type
    - For instance, the *Certificate Type* Attribute only exists for Certificate objects
  - Some Attributes are implicitly set by certain operations
    - Certificate Type is implicitly set by Register, Certify, and Re-certify
  - Client can set explicitly some of the Attributes
    - Certificate Type cannot be set by the client
  - Not all Attributes can be added, or subsequently modified or deleted once set
    - Certificate Type cannot added, modified or deleted
  - Some Attributes can have multiple values (or instances) organized with indices
    - For instance, a Symmetric Key object may belong to multiple groups, hence its *Object Group* Attribute will have multiple values

KMIP Attributes cont’d

- 31 Attributes defined



Describes what “is” the object

- Unique Identifier
- Name
- Object Type
- Cryptographic Algorithm
- Cryptographic Length
- Cryptographic Parameters
- Certificate Type
- Certificate Issuer
- Certificate Subject
- Digest

Describes how to “use” the object

- Operation Policy Name
- Cryptographic Usage Mask
- Lease Time
- Usage Limits
- State
- Initial Date
- Activation Date
- Process Start Date
- Protect Stop Date
- Deactivation Date
- Destroy Date
- Compromise Occurrence Date
- Compromise Date
- Revocation Reason
- Archive Date

Describes other features of the object

- Object Group
- Link
- Application Specific ID
- Contact Information
- Last Change Date
- Custom Attribute

Key Lifecycle States and Transitions

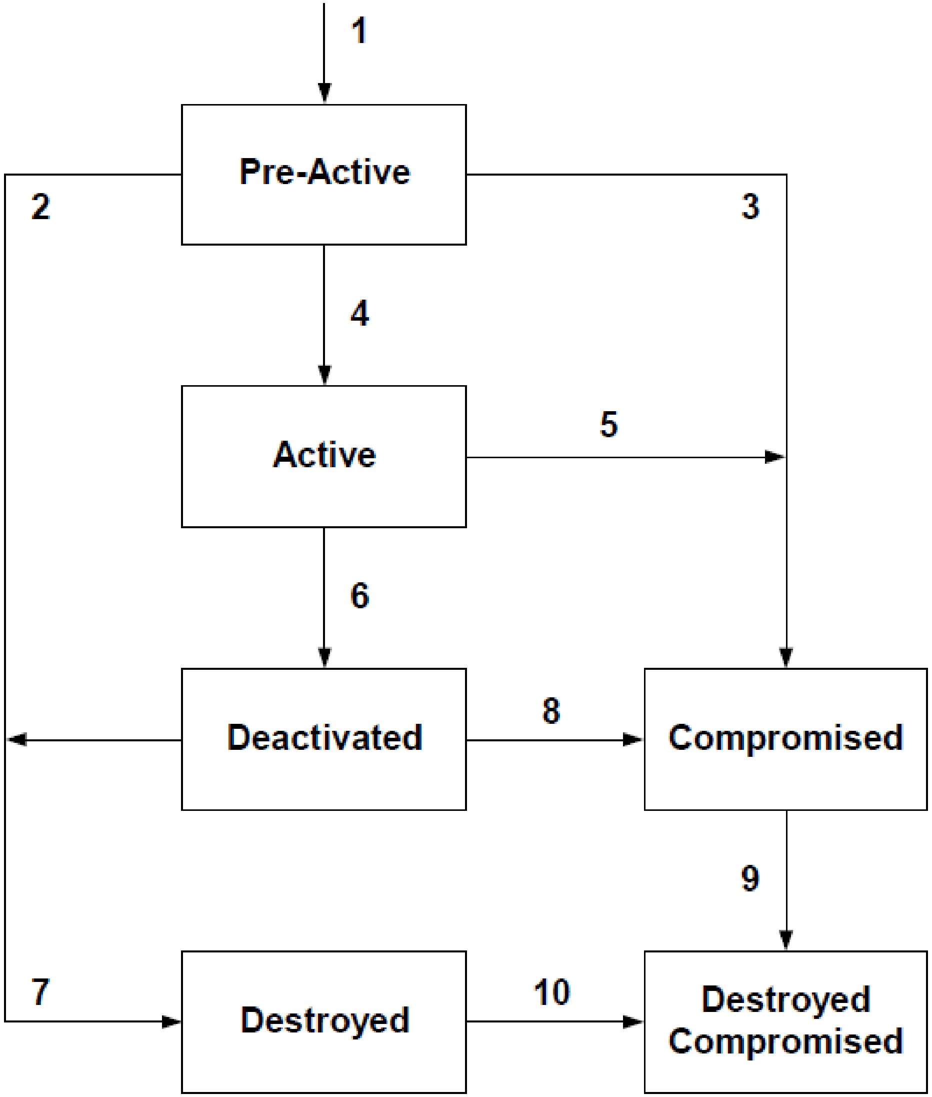
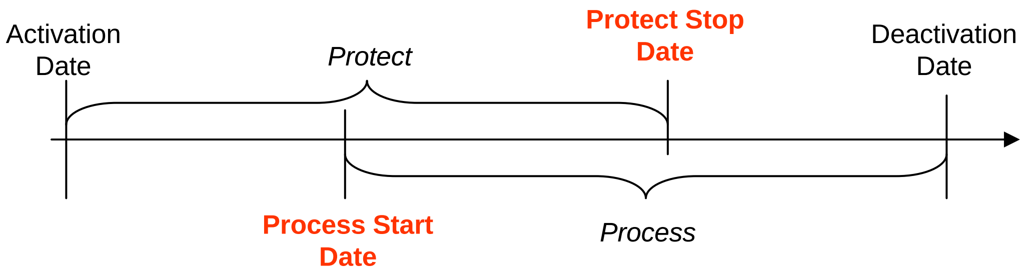


Illustration of the Lifecycle Dates

Symmetric key:



Activation Date

Protect

Protect Stop Date

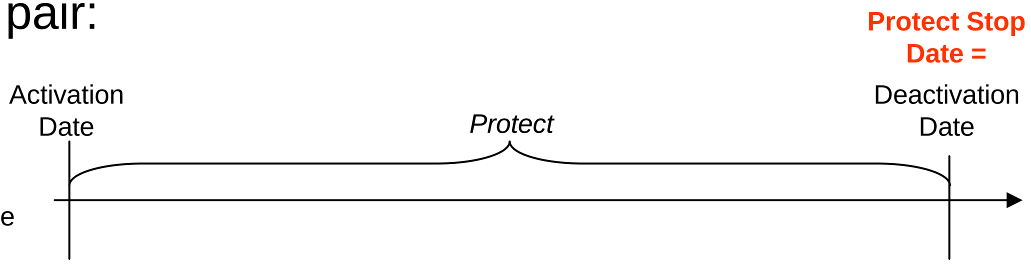
Deactivation Date

Process Start Date

Process

Asymmetric key pair:

Public key for encryption
Private key for digital signature



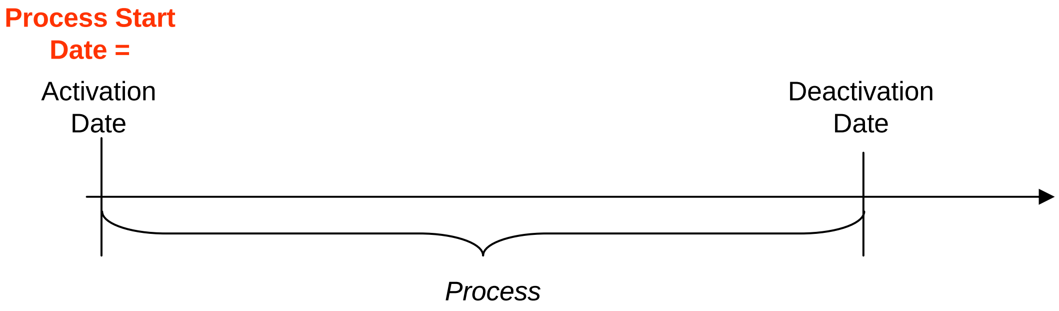
Protect Stop Date =

Activation Date

Protect

Deactivation Date

Private key for decryption
Public key for digital signature



Process Start Date =

Activation Date

Deactivation Date

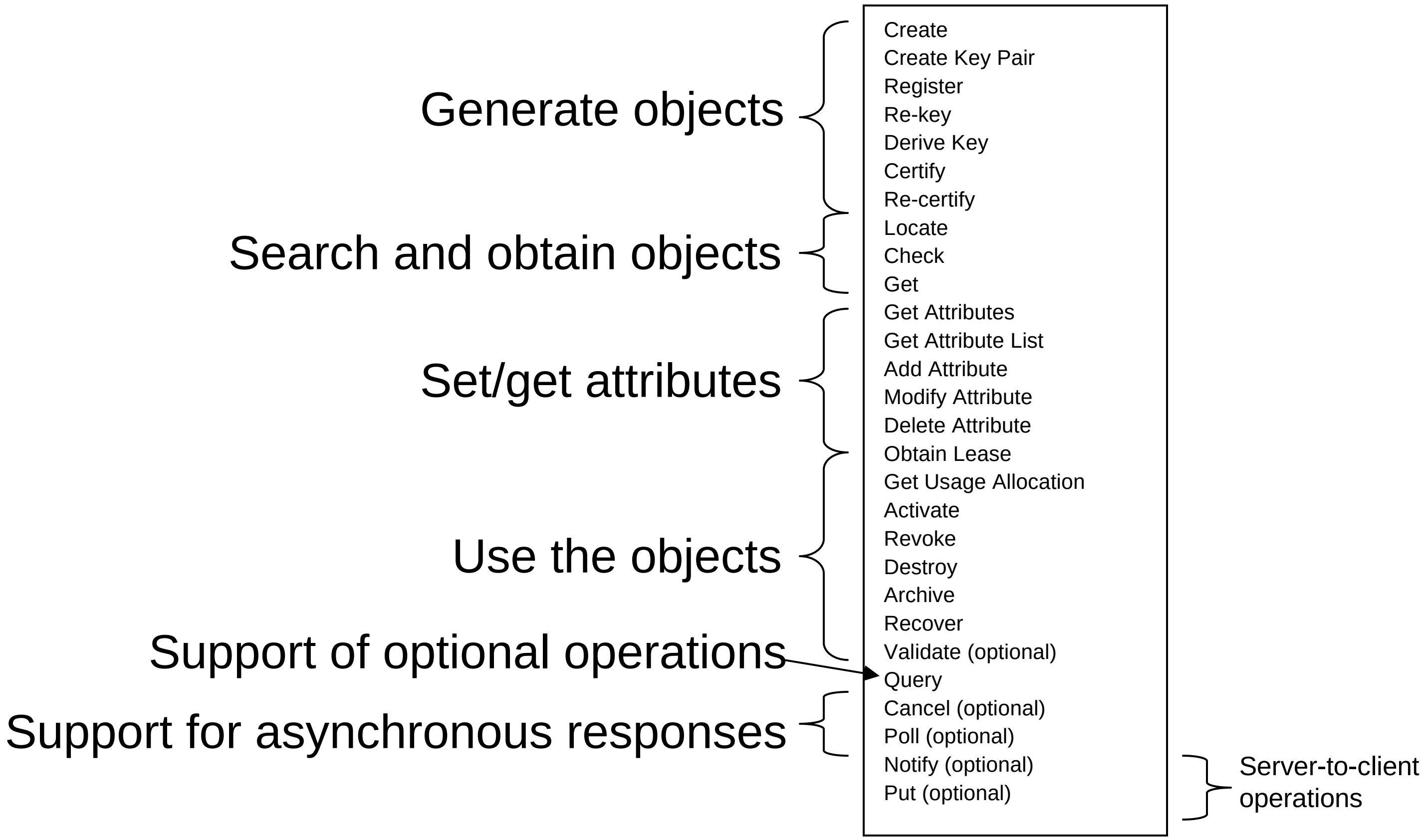
Process

Client-to-server Operations

- Operation consists of a request from client followed by server response
- Multiple operations can be batched in a single request-response pair
  - ID Placeholder can be used to propagate the value of the object’s Unique Identifier among operations in the same batch
  - Can be used to implement atomicity
- Requests may contain Template-Attribute structures with the desired values of certain attributes
- Responses contain the attribute values that have been set differently than as requested by the client

Client-to-server Operations cont’d

- 26 client-to-server operations defined



Generate objects

Search and obtain objects

Set/get attributes

Use the objects

Support of optional operations

Support for asynchronous responses

Create
Create Key Pair
Register
Re-key
Derive Key
Certify
Re-certify
Locate
Check
Get
Get Attributes
Get Attribute List
Add Attribute
Modify Attribute
Delete Attribute
Obtain Lease
Get Usage Allocation
Activate
Revoke
Destroy
Archive
Recover
Validate (optional)
Query
Cancel (optional)
Poll (optional)
Notify (optional)
Put (optional)

Server-to-client operations

Server-to-client Operations

- Unsolicited messages from the server to the client with the following operations:
  - Notify operation, used by server to inform client about attribute-value changes
  - Push operation, used by server to provide an object and attributes to client, indicating whether the new object is replacing an existing object or not
  - Batching can be used
  - Support is optional

Message Contents and Format

- Protocol messages consist of requests and responses, each with a header and one or more batch items with operation payloads and message extensions
- Header:
  - Protocol version
  - Maximum response size (optional, in request)
  - Time Stamp (optional in request, required in response)
  - Authentication (optional)
  - Asynchronous Indicator (optional, in request, no support for asynchronous response is default)
  - Asynchronous Correlation Value (optional, in response). Used later on for asynchronous polling
  - Result Status: Success, Pending, Undone, Failure (required, in response)
  - Result Reason (required in response if Failure, optional otherwise)
  - Result Message (optional, in response)
  - Batch Order Option (optional, in request, in-order processing is default). Support at server is optional
  - Batch Error Continuation Option: Undo, Stop, Continue. Stop (optional, in request, Stop is default). Support at server is optional
  - Batch Count
- Batch Item:
  - Operation (enumeration)
  - Unique Message ID (required if more than one batch item in message)
  - Payload (the actual operation request or response)
  - Message Extension (optional, for vendor-specific extensions)

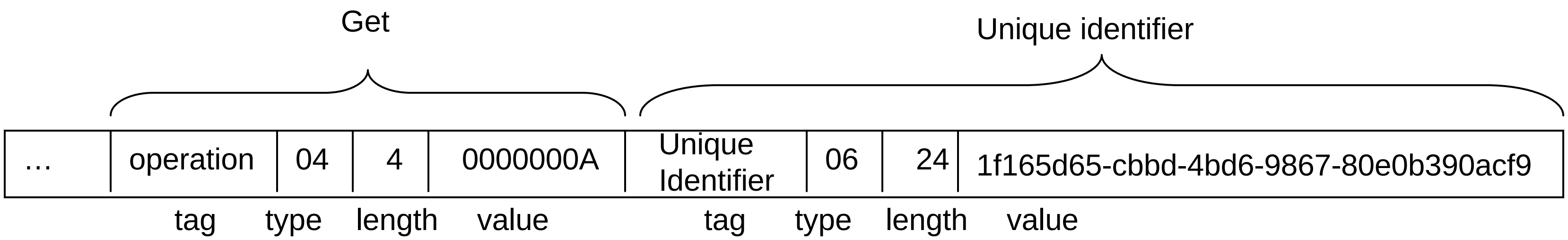
Message Encoding

- Example of TTLV encoding of the *Application Specific ID* Attribute
  - Attribute identified by its name “Application Specific ID”
  - Shows value at index 2

<table>
<tr><th>Tag</th><th>Type</th><th>Length</th><th>Value</th></tr>
<tr><td>Attribute</td><td>Structure</td><td>&lt;varies&gt;</td><td>
<table>
<tr><th>Tag</th><th>Type</th><th>Length</th><th>Value</th></tr>
<tr><td>Attribute Name</td><td>String</td><td>&lt;varies&gt;</td><td>“Application Specific ID”</td></tr>
<tr><td>Attribute Index</td><td>Integer</td><td>4</td><td>2</td></tr>
<tr><td>Attribute Value</td><td>Structure</td><td>&lt;varies&gt;</td><td>
<table>
<tr><th>Tag</th><th>Type</th><th>Length</th><th>Value</th></tr>
<tr><td>App. Name</td><td>String</td><td>&lt;varies&gt;</td><td>“ssl”</td></tr>
<tr><td>App. ID</td><td>String</td><td>&lt;varies&gt;</td><td>“www.example.com”</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Message Encoding cont'd

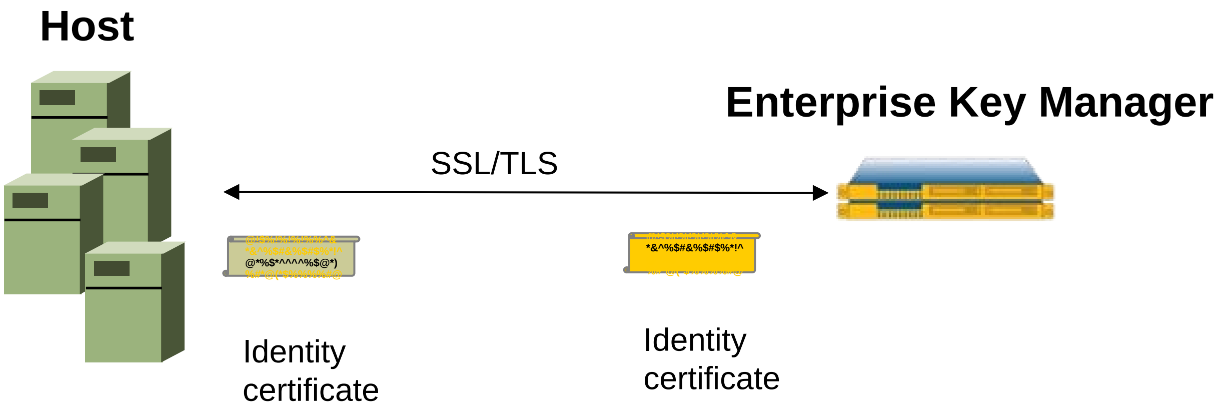
- In a TTLV-encoded message, Attributes are identified either by tag value or by their name (see previous slide), depending on the context:
  - When the operation lists the attribute name among the objects part of the request/response (such as Unique Identifier), its tag is used in the encoded message
  - When the operation does not list the attribute name explicitly, but instead includes Template-Attribute (such as in the Create operation) or Attribute (such as in Add Attribute) objects as part of the request/response, its name is used in the encoded message



	Get				Unique identifier			
…	operation	04	4	0000000A	Unique Identifier	06	24	1f165d65-cbbd-4bd6-9867-80e0b390acf9
	tag	type	length	value	tag	type	length	value

Authentication

- Authentication is external to the protocol
- All servers should support at least
  - SSL/TLS
  - https
- Authentication message field contains the Credential Base Object
  - Client or server certificate in the case of SSL/TLS or https



KMIP
Use Cases

http://xml.coverpages.org/KMIP/KMIP-UseCases-v0.98-final.pdf

KMIP Use Cases

- Purpose: provide examples of message exchanges for common use cases
- Categories
  - basic functionality (create, get, register, delete of sym. keys and templates)
  - life-cycle support (key states)
  - auditing and reporting
  - key exchange
  - asymmetric keys
  - key roll-over
  - archival
  - vendor-specific message extensions
- Details of the message composition and TTLV encoding (encoded bytes included)

KMIP Use Cases: Example

- Request containing a Get payload

The operation (object type) and payload parameter

```
Get (symmetric key)
In: uuidKey
```

Fields and structure of the message (length not shown)

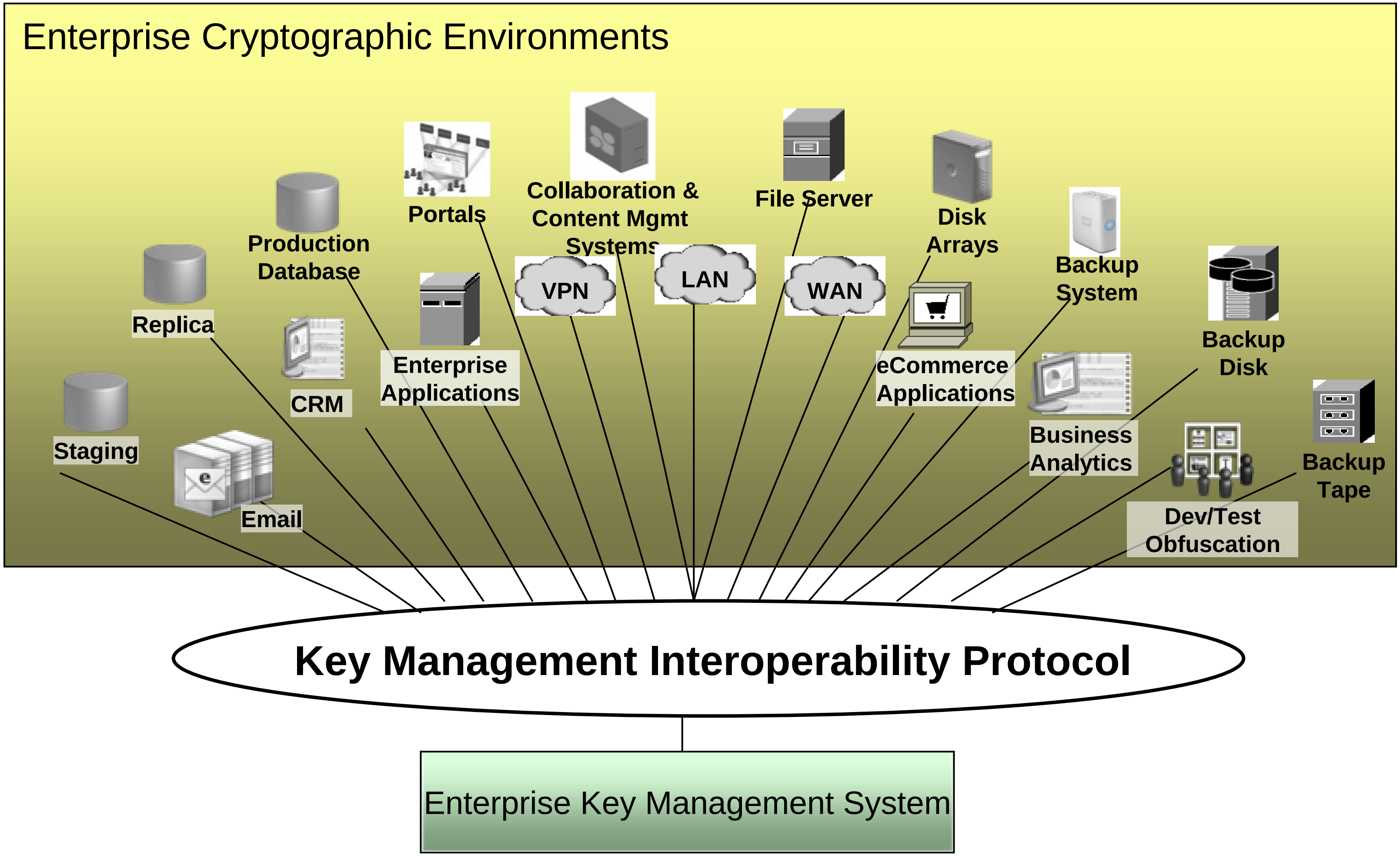
```
Tag: Request Message (0x42000073), Type: Structure (0x80), Data:
  Tag: Request Header (0x42000072), Type: Structure (0x80), Data:
    Tag: Protocol Version (0x42000065), Type: Structure (0x80), Data:
      Tag: Protocol Version Major (0x42000066), Type: Integer (0x01), Data: 0x00000000 (0)
      Tag: Protocol Version Minor (0x42000067), Type: Integer (0x01), Data: 0x00000062 (98)
    Tag: Batch Count (0x4200000D), Type: Integer (0x01), Data: 0x00000001 (1)
  Tag: Batch Item (0x4200000F), Type: Structure (0x80), Data:
    Tag: Operation (0x42000057), Type: Enumeration (0x04), Data: 0x0000000A (Get)
    Tag: Request Payload (0x42000074), Type: Structure (0x80), Data:
      Tag: Unique Identifier (0x4200008F), Type: Text String (0x06), Data:
          96789141-62bf-4352-b1c4-9d48dac4b77d
```

TTLV byte encoding of the message

```
4200007380000000854200007280000000304200006580000001A42000066010000000400000000420000670100000004000000062
```

Conclusion

KMIP: enabling enterprise key management through standard protocol



Enterprise Cryptographic Environments

Production Database

Portals

Collaboration & Content Mgmt Systems

File Server

Disk Arrays

Backup System

Backup Disk

Backup Tape

Replica

Staging

CRM

Email

Enterprise Applications

VPN

LAN

WAN

eCommerce Applications

Business Analytics

Dev/Test Obfuscation

Key Management Interoperability Protocol

Enterprise Key Management System

Q & A