Interoperable Key Management using the OASIS KMIP Standard

Agenda

The Need for Interoperable Key Management

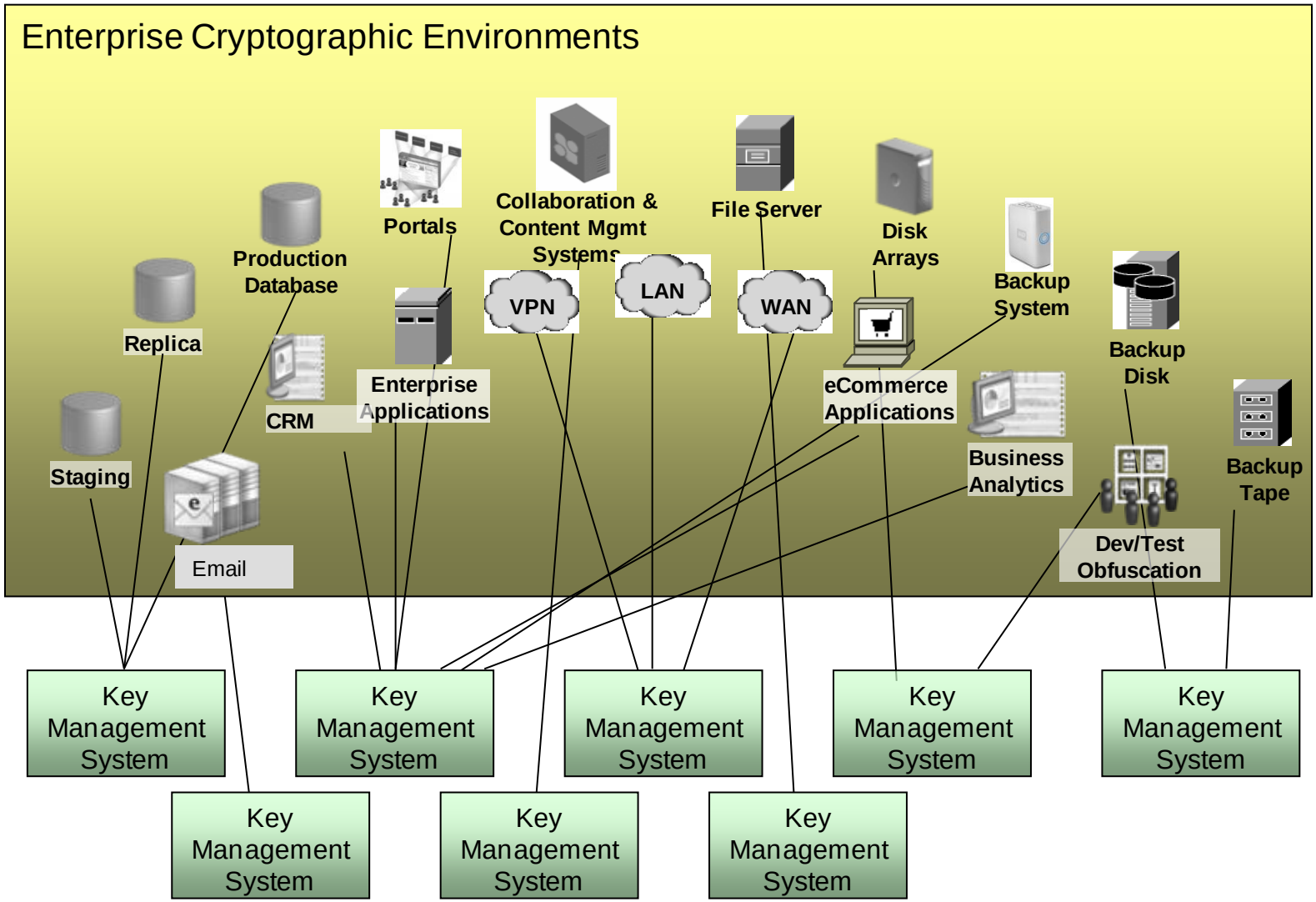
KMIP Overview

KMIP Specification

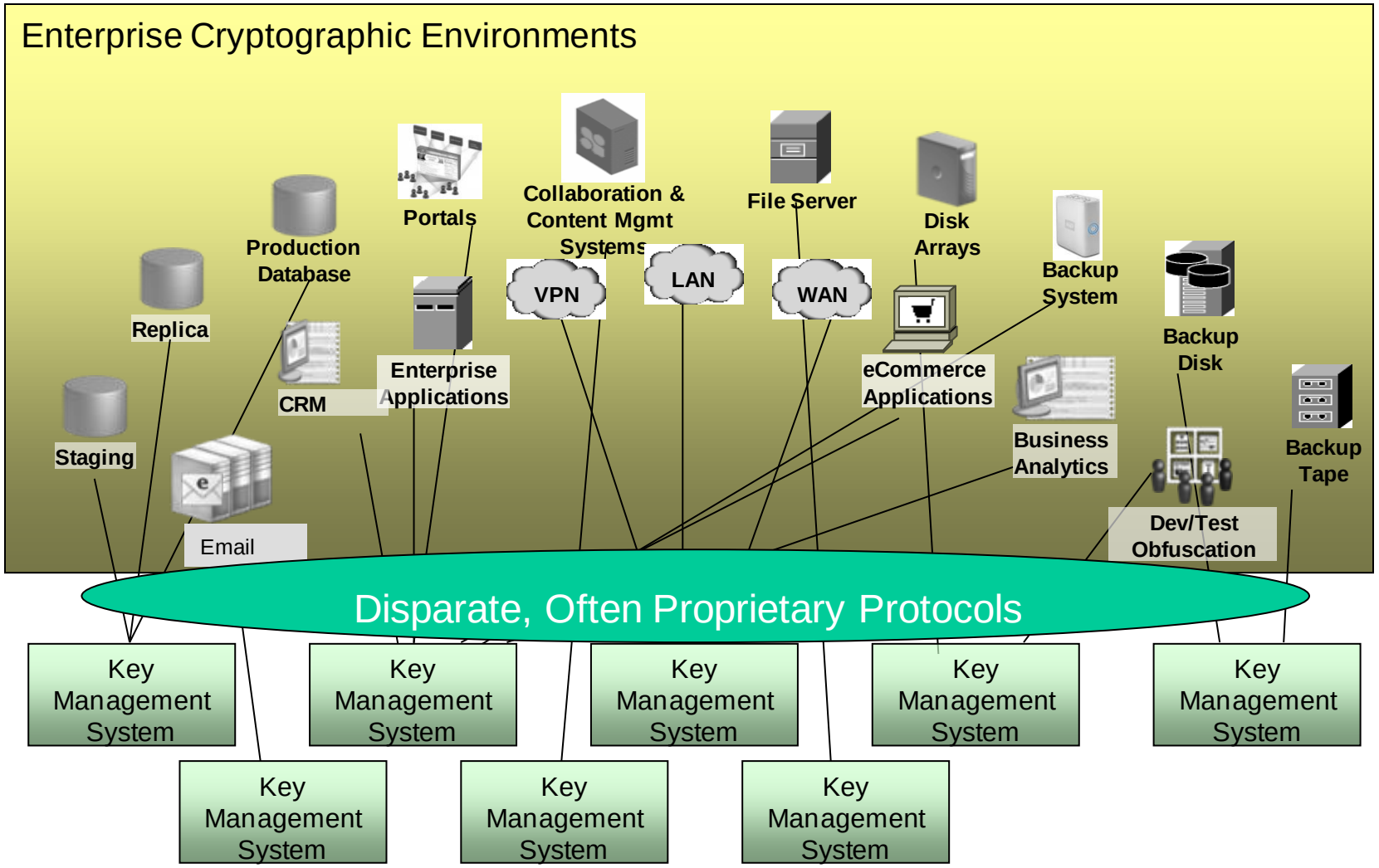
The Need for Interoperable Key Management

- 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

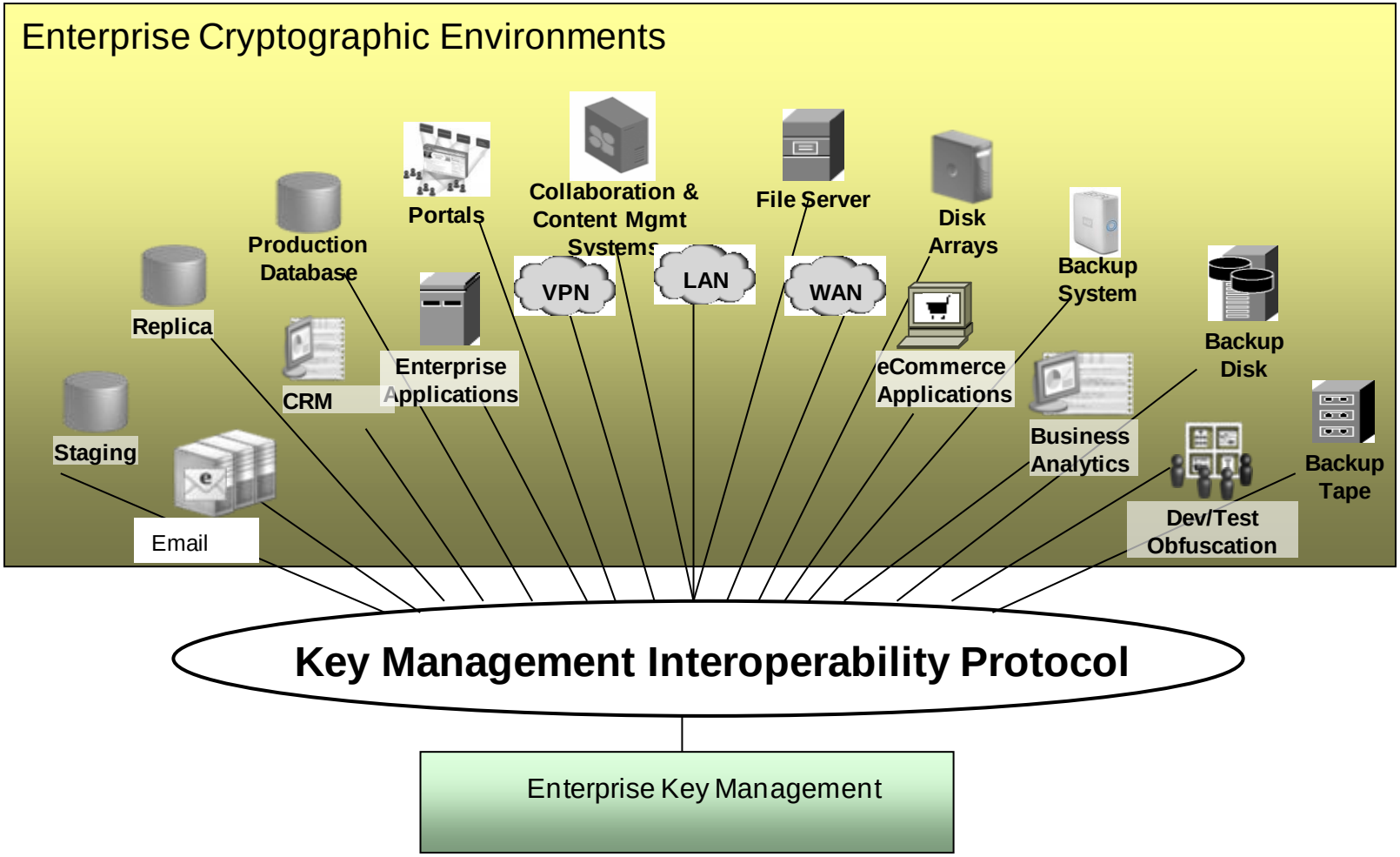


Often, Each Cryptographic Environment Has Its Own Protocol



KMIP Overview

KMIP: Single Protocol Supporting Enterprise Cryptographic Environments



OASIS KMIP Technical Committee

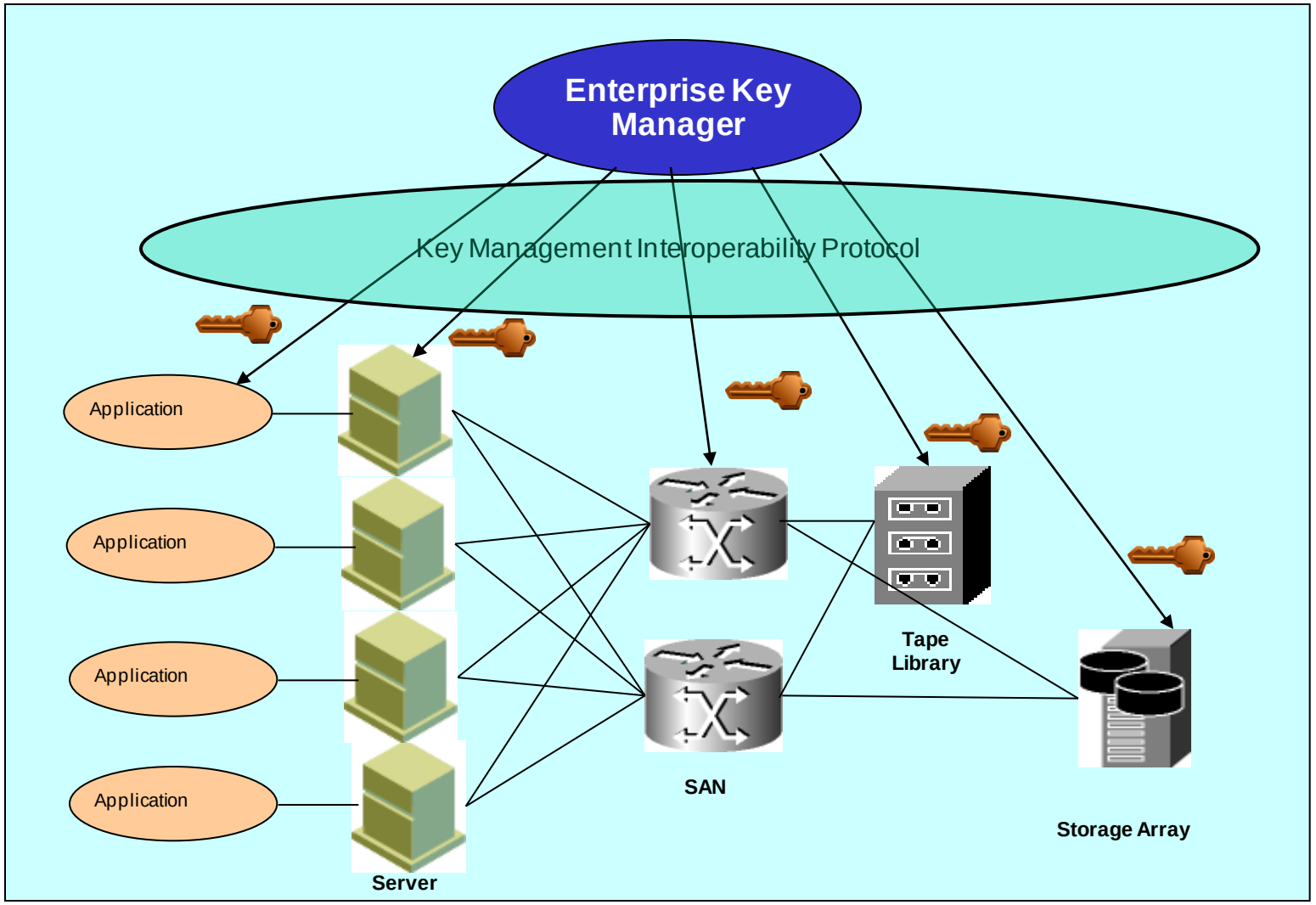
OASIS (Organization for the Advancement of Structured Information Standards) is a not-for-profit consortium that drives the development, convergence and adoption of open standards for the global information society.

KMIP Technical Committee chartered in March 2009.

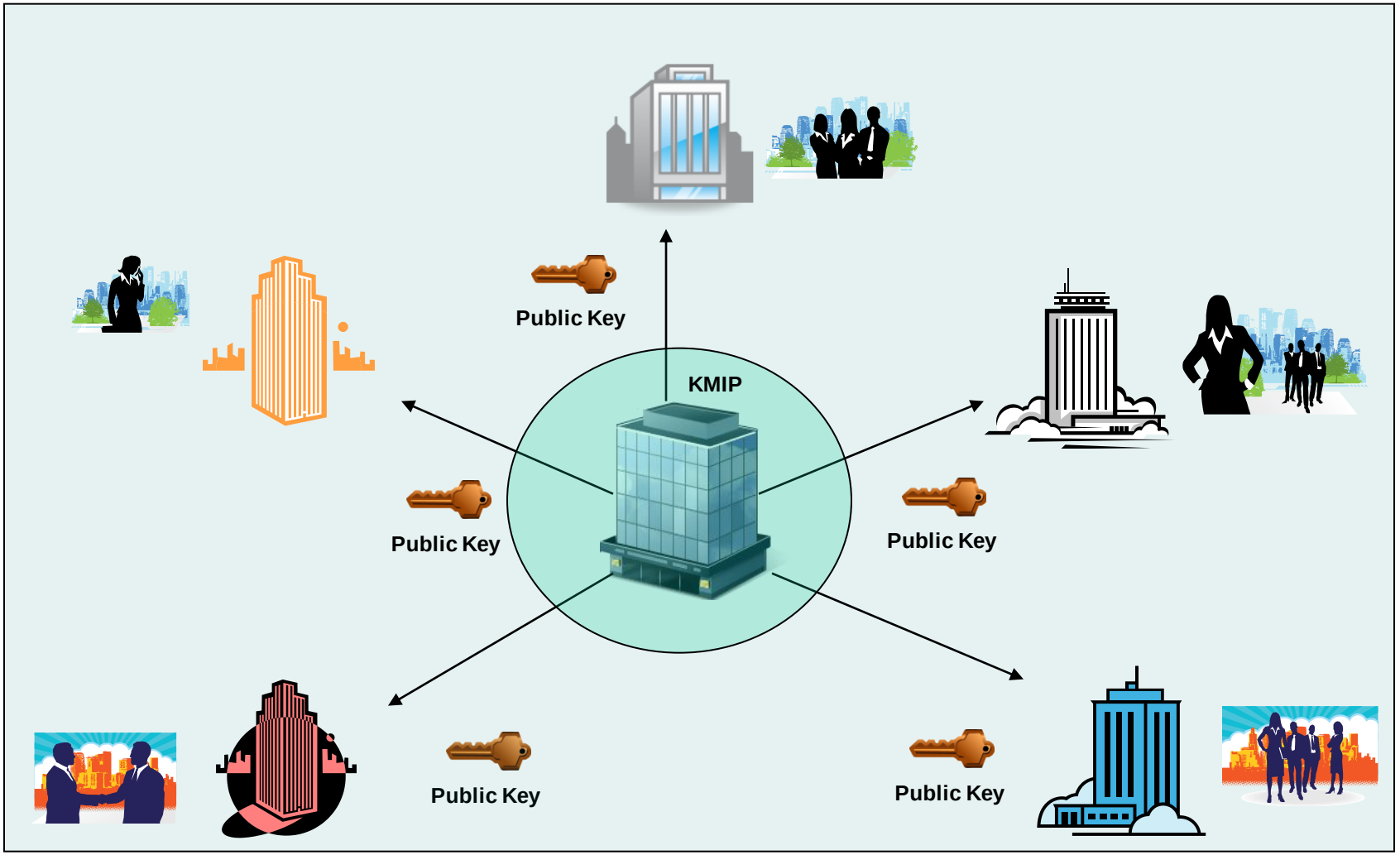
> "The KMIP TC will develop specification(s) for the interoperability of Enterprise Key Management (EKM) services with EKM clients. The specifications will address anticipated customer requirements for key lifecycle management (generation, refresh, distribution, tracking of use, life-cycle policies including 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."

- I.P. mode: "R.F. on RAND"

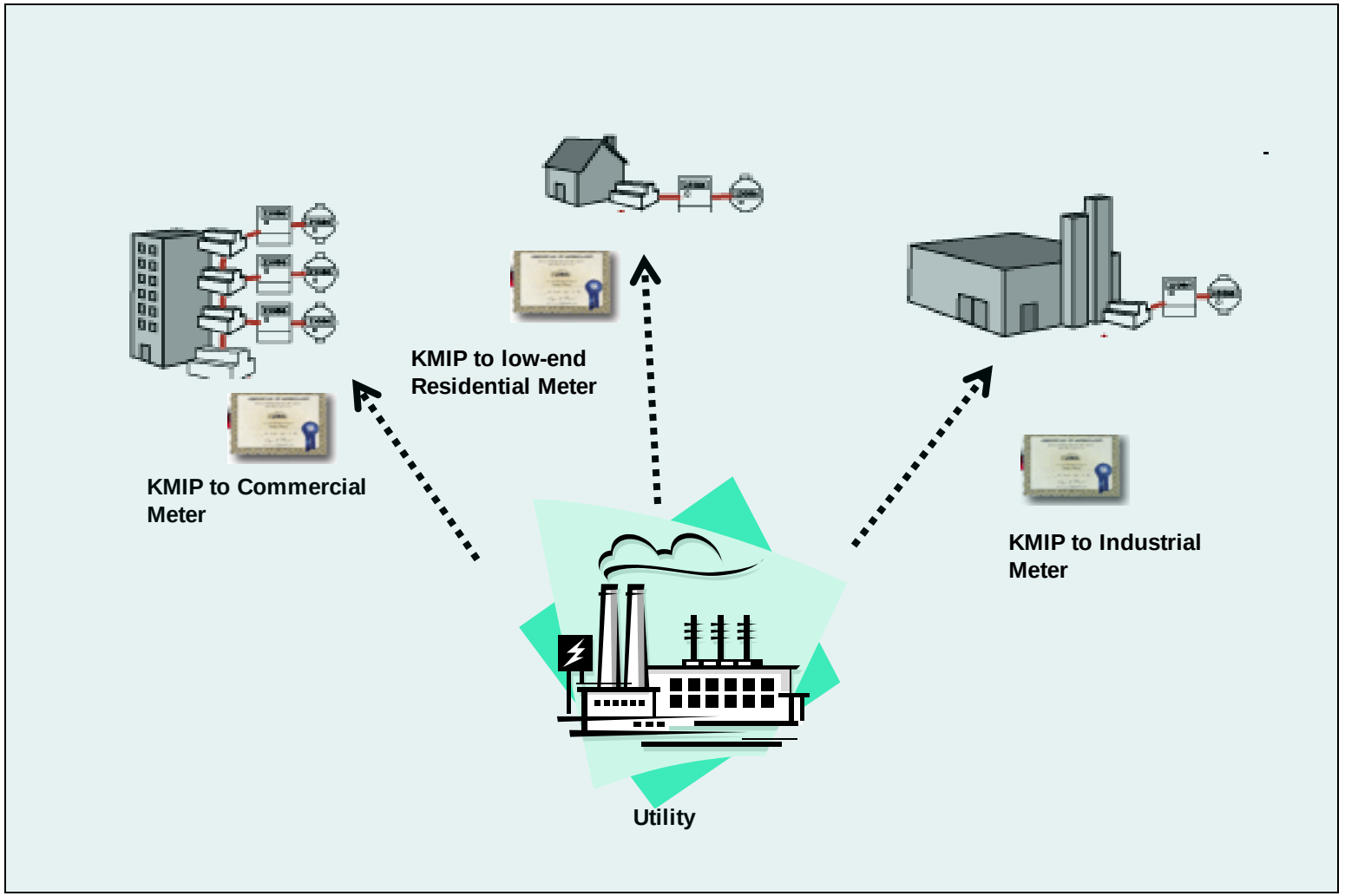
KMIP Objects, Operations and Attributes: Symmetric Encryption Keys



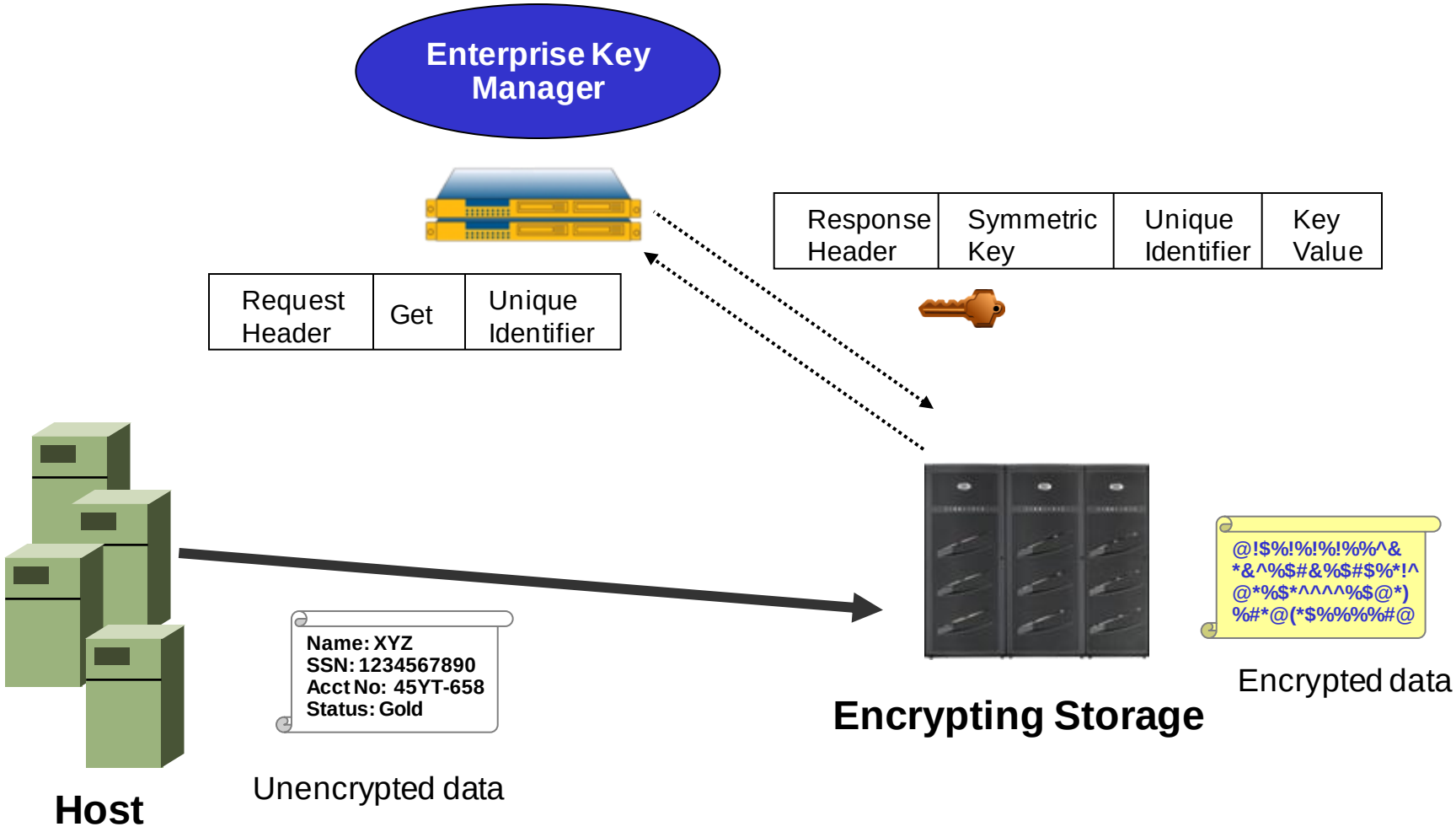
KMIP Objects, Operations and Attributes: Asymmetric Keys



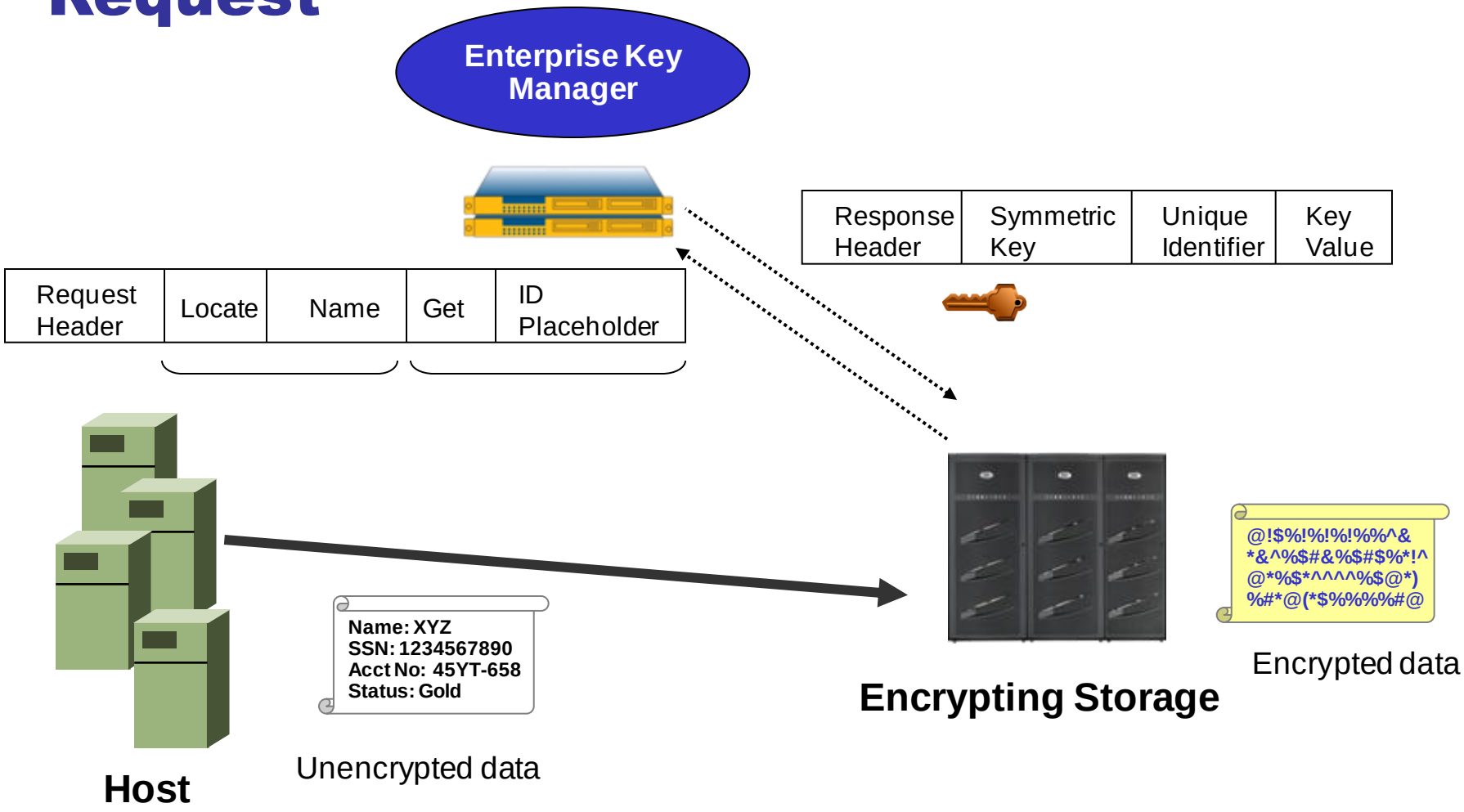
KMIP Objects, Operations and Attributes: Digital Certificates



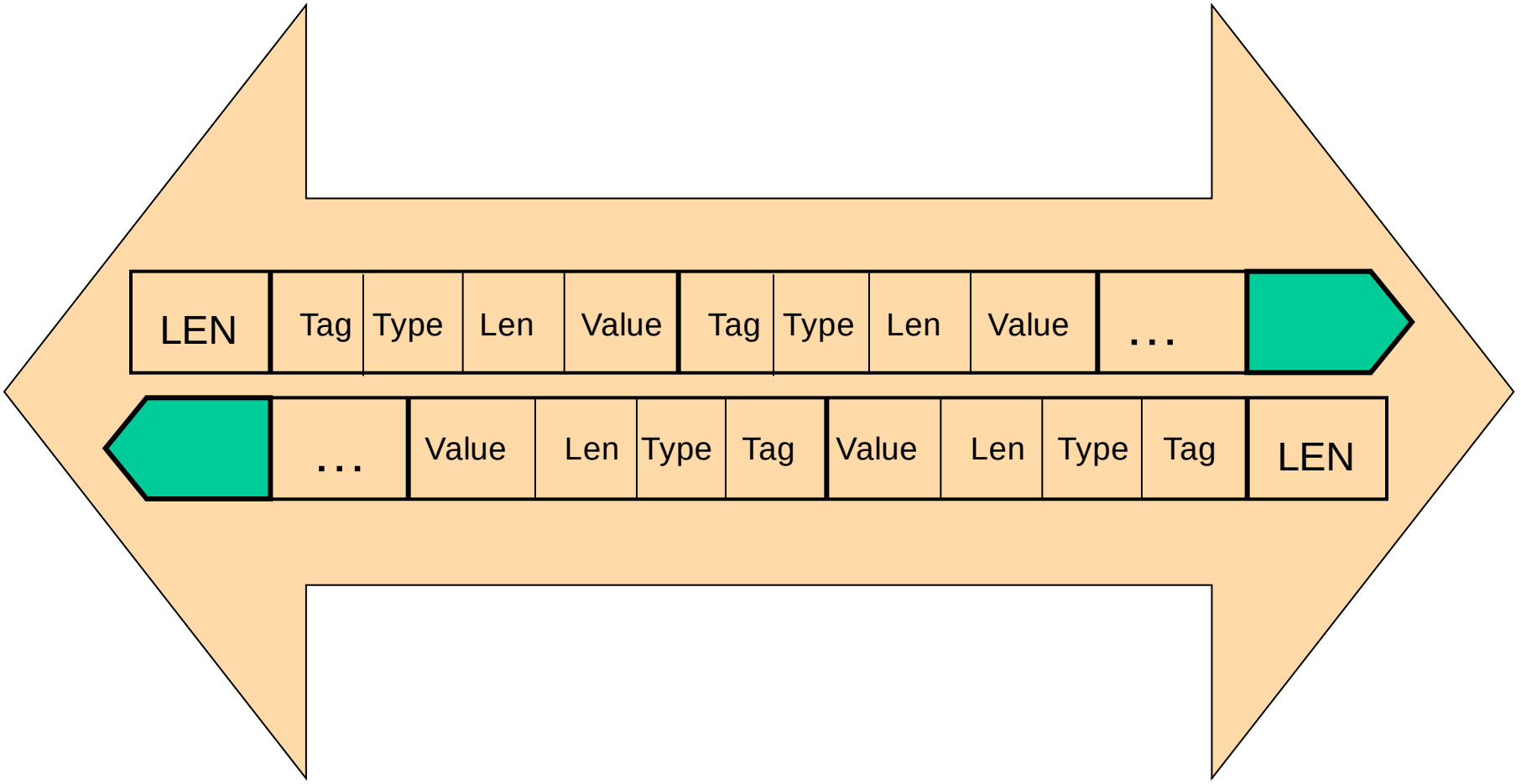
KMIP Request / Response Model



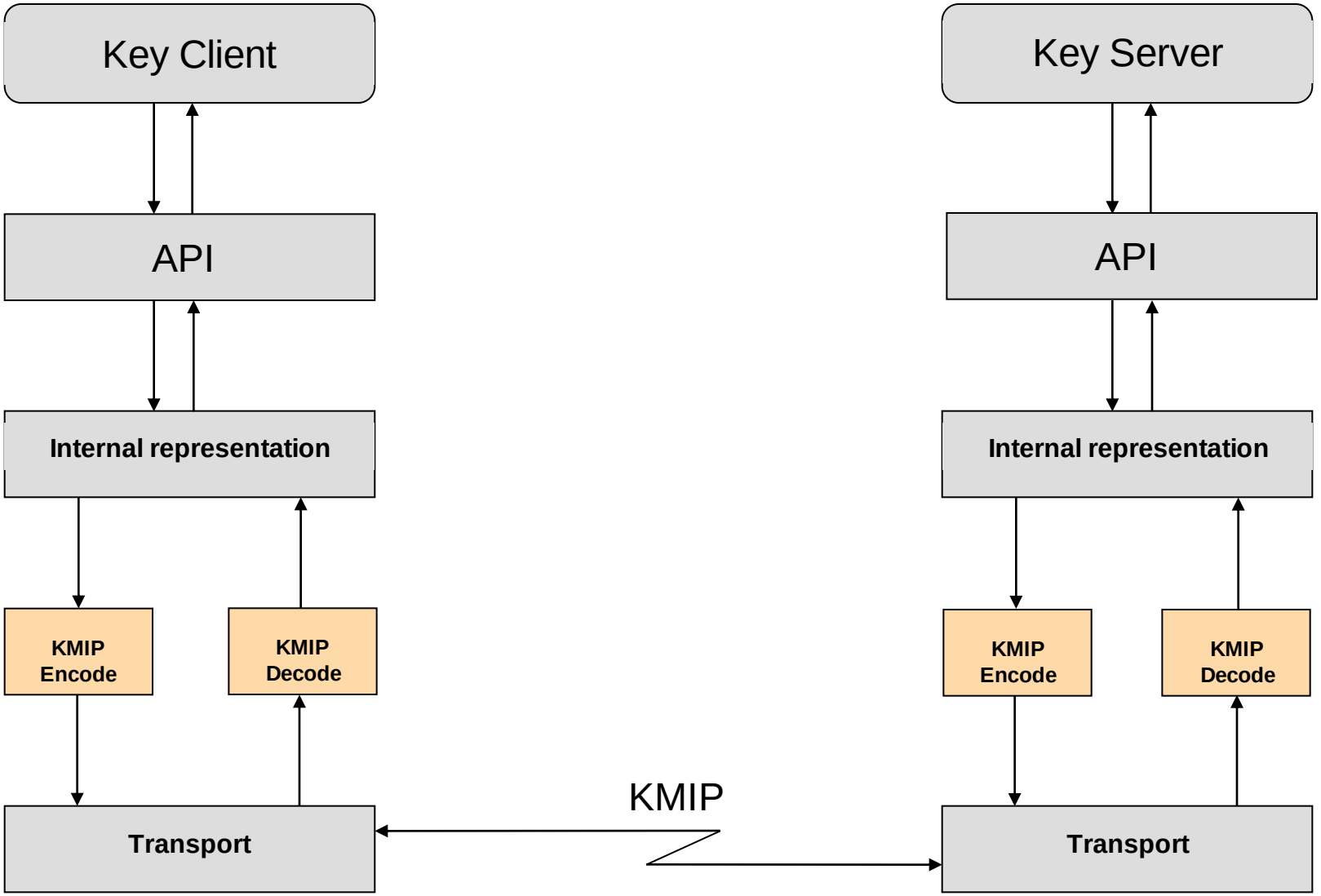
Supporting Multiple Operations per Request



Messages in TTLV Format

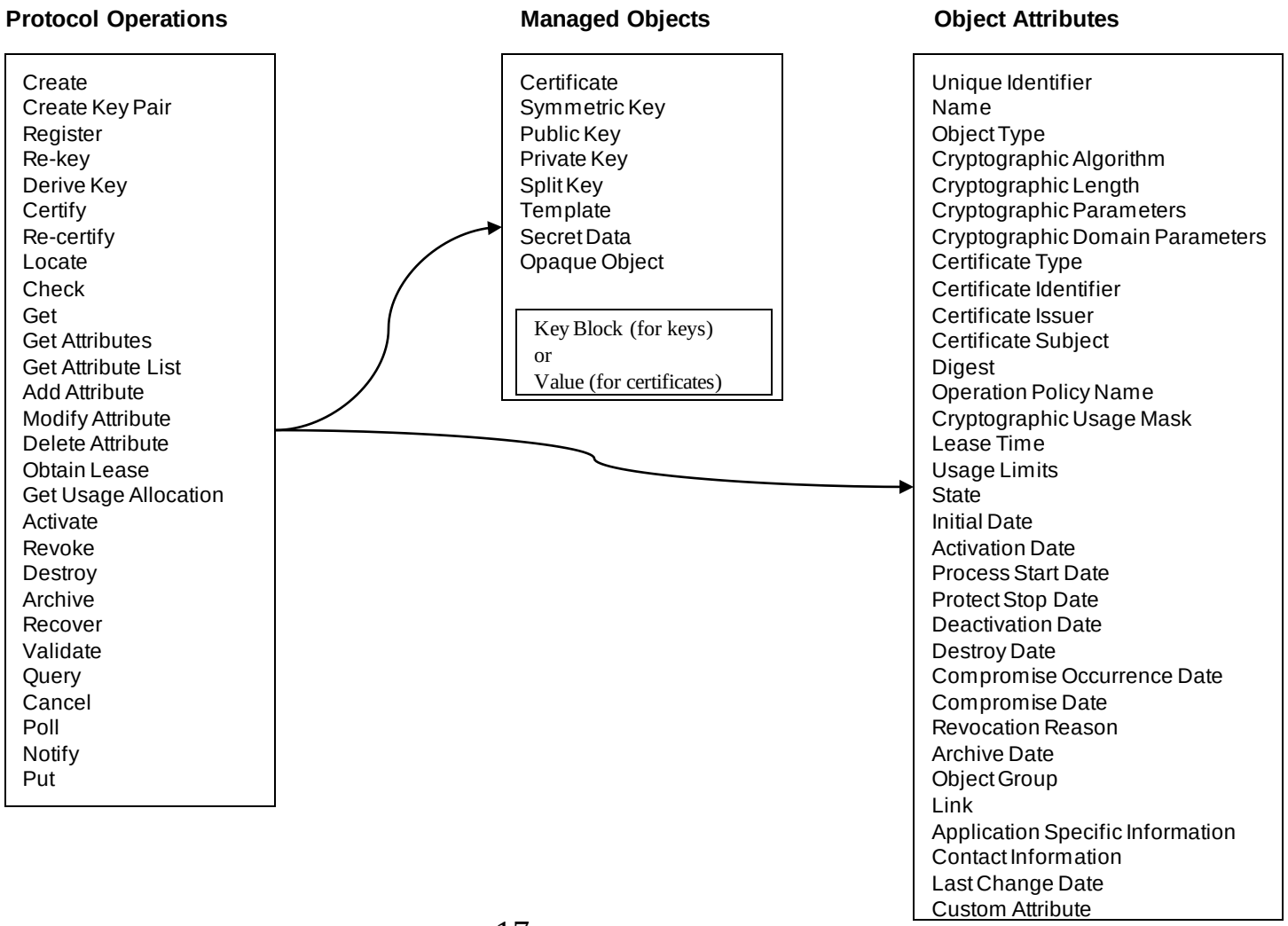


Transport-Level Encoding



KMIP Specification

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

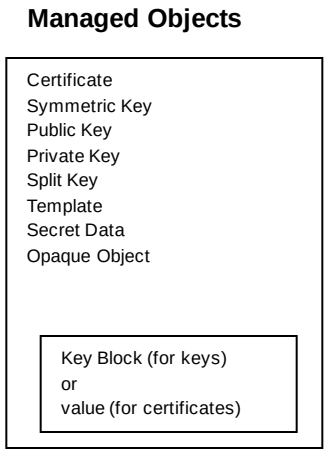


Base Objects

- 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

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



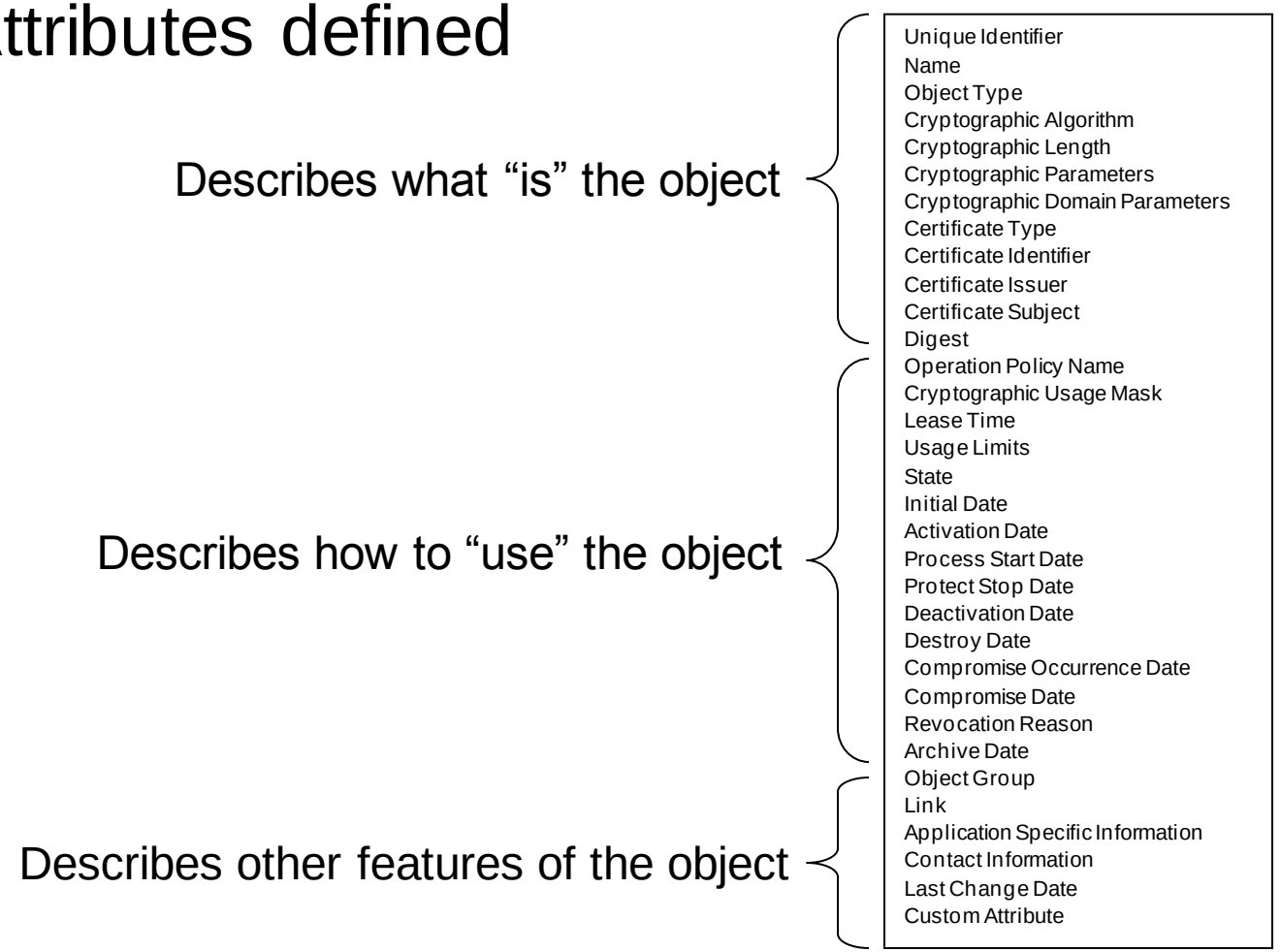
- Managed Objects
  - Template
    - Template has a subset of Attributes that indicate **what an object** created from such a template **is**
  - Opaque Object, without Key Block

Attributes

- Attributes contain the “metadata” 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

Attributes - 2

- 33 Attributes defined



Key Lifecycle States and Transitions

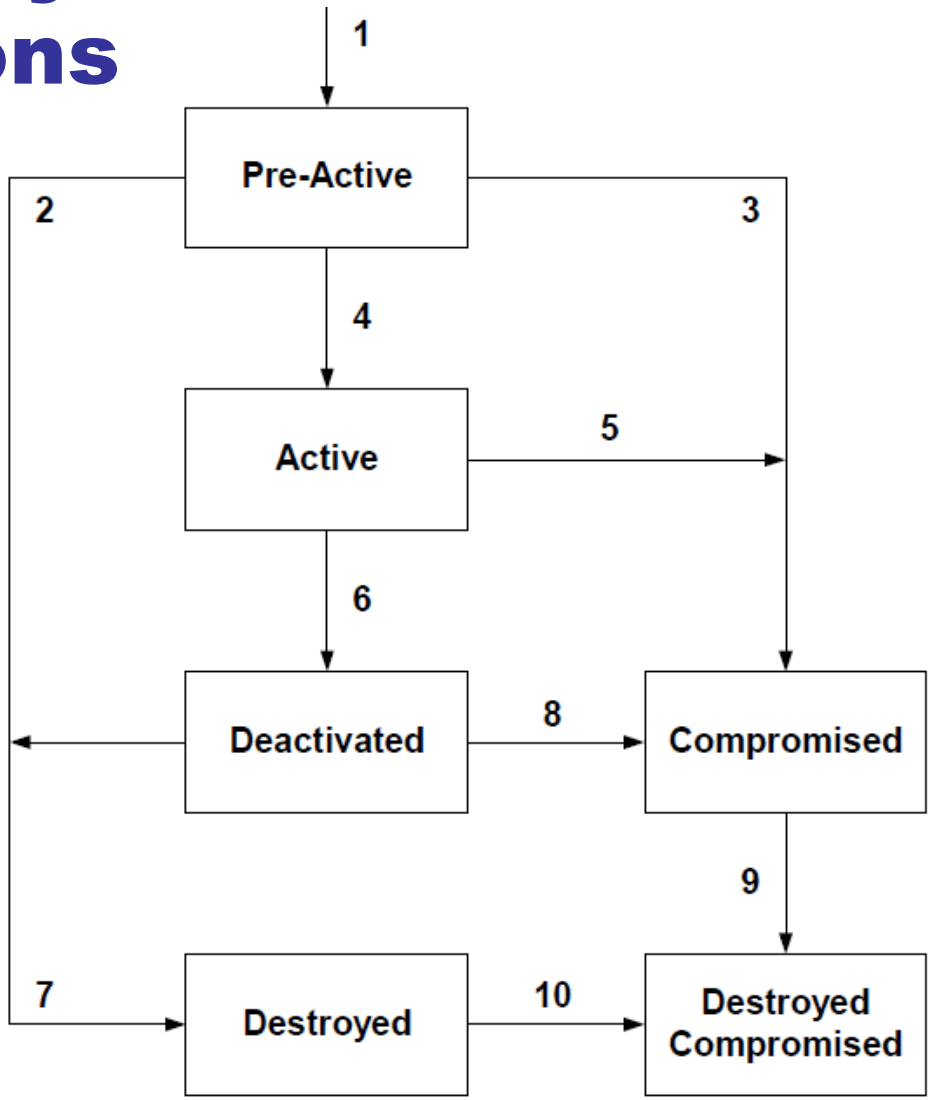
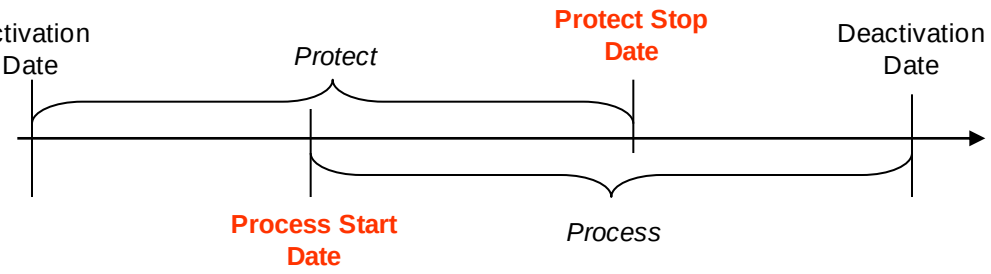


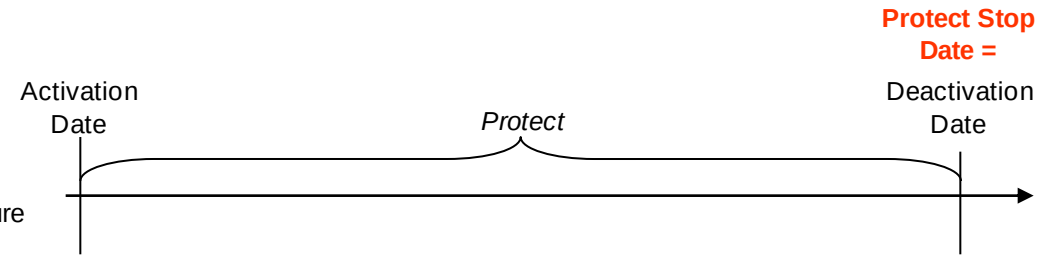
Illustration of the Lifecycle Dates

Symmetric key:

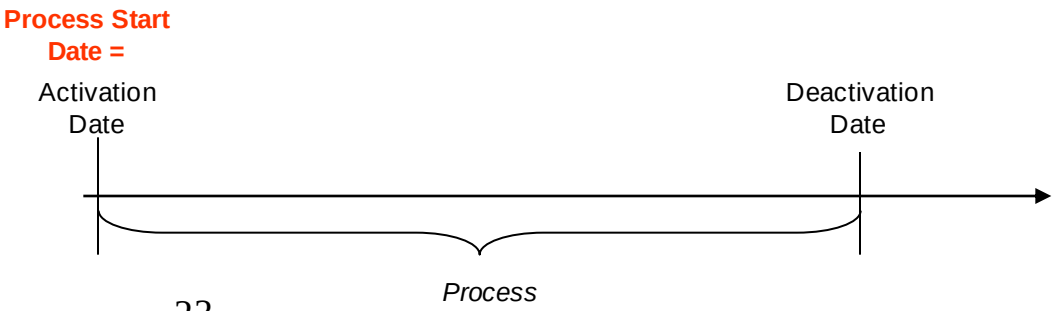


Asymmetric key pair:

Public key for encryption
Private key for digital signature



Private key for decryption
Public key for digital signature

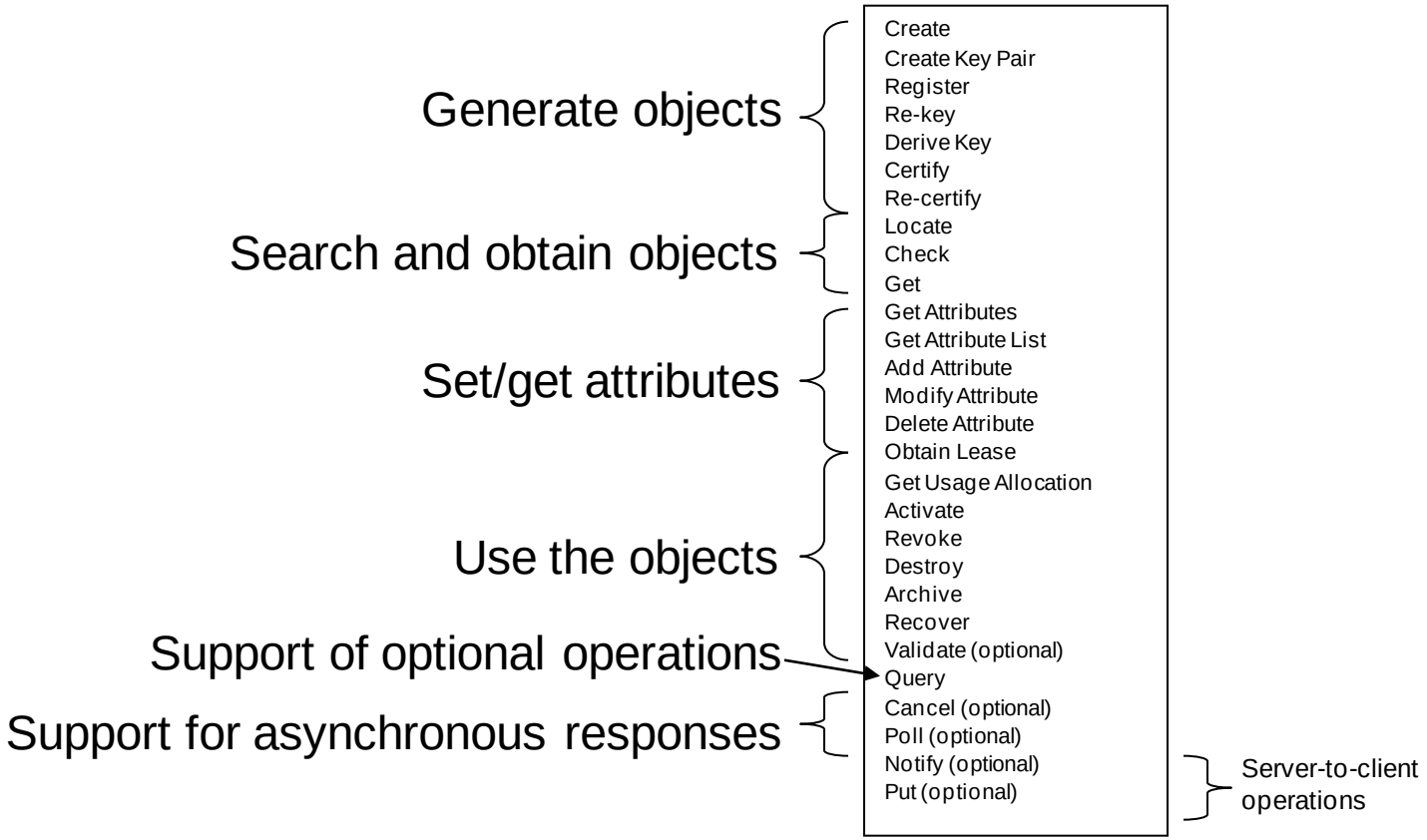


Client-to-server Operations

- 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 - 2

- 26 client-to-server operations defined



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

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 Information* Attribute
  - Attribute identified by its name "Application Specific Information"
  - Shows value at index 2

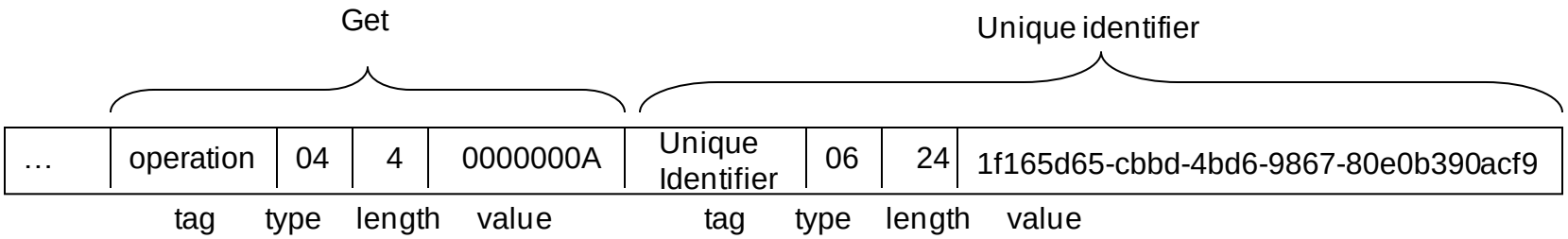
Tag	Type	Length	Value
Attribute	Structure	<varies>	

Tag	Type	Length	Value
Attribute Name	String	<varies>	"Application Specific Information"
Attribute Index	Integer	4	2
Attribute Value	Structure	<varies>	

Tag	Type	Length	Value
App. Name	String	<varies>	"ssl"
App. ID	String	<varies>	"www.example.com"

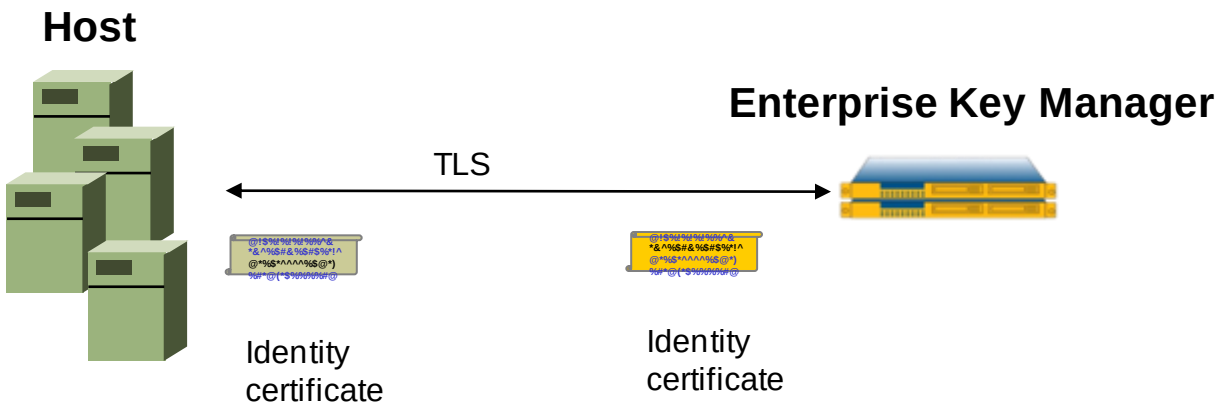
Message Encoding - 2

- 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



Authentication

- Authentication is external to the protocol
- All servers should support at least
  - TLS 1.2
- Authentication message field contains the Credential Base Object
  - Allows inclusion of additional credential information



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)

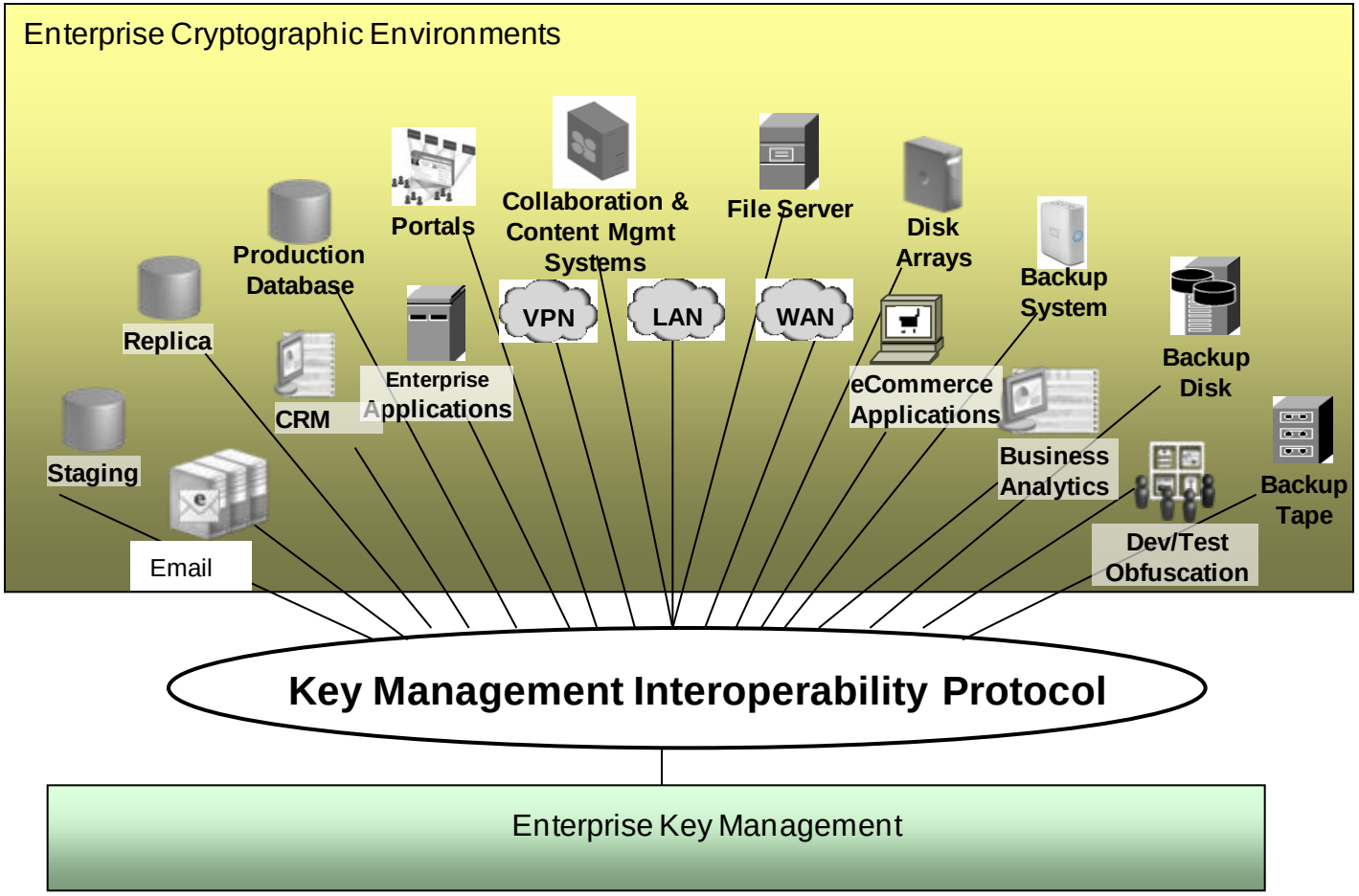
```
Tag: Request Message (0x42000073), Type: Structure (0x80), Data:
  Tag: Request Header (0x42000072), Type: Structure (0x80), Data:
    Tag: Protocol Version (0x42000065), Type: Structure (0x80), Data:
      Tag: Protocol Version Major (0x42000066), Type: Integer (0x01), Data: 0x00000000 (0)
      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

```
42000073800000008542000072800000003042000065800000001A42000066010000000400000000420000670100000004000000620200000D0100000004000000014200000F80000000434200005704000000040000000A4200007480000000 2D4200008F0600000024393637383931343 12D363262662D343335322D623163342D396434386461633462373764
```

Conclusion

KMIP: enabling enterprise key management through standard protocol.



Questions?