Key Management Interoperability Protocol (KMIP)

Addressing the Need for Standardization in Enterprise Key Management

Version 1.0, May 20, 2009

Table of Contents

Executive Summary

The increasing use of encryption, certificate-based device authentication, asymmetric key pairs and digital signature reflects the critical importance of cryptography in addressing regulatory requirements, protecting intellectual property and controlling the exposure of sensitive information. However, the widespread use of these and other cryptographic technologies is complicated by inconsistencies and duplication in the key management systems supporting the applications, devices and systems using these technologies.

For example, each native tape encryption system tends to have its own key management system, separate from the key management system for application encryption, or database encryption, or file encryption. Full-disk encryption systems for laptops have their own key management systems, as do encryption systems for disk-array storage environments and content management systems. Asymmetric key pairs and digital certificates similarly have their own key management systems as well. This proliferation of key management systems results in higher operational and infrastructure costs for enterprises using encryption, certificates, asymmetric key pairs and other cryptographic technologies.

Even in those cases where a single key management system can support multiple types of security objects and multiple kinds of cryptographically-enabled systems, there are typically different communication protocols between the key management servers and each of the cryptographic clients that communicate with it. An enterprise key management system, for example, is likely to have to communicate with an encrypting tape drive using a communication protocol specific to that tape drive, or with a SAN switch by means of a communication protocol specific to that switch, or with an application requiring asymmetric keys with yet another communication protocol and so on. This proliferation of protocols, even when supported by a single enterprise key manager, results in higher costs for developing and supporting the key manager; costs that ultimately get passed on to the enterprises deploying security solutions.

The Key Management Interoperability Protocol (KMIP), recently introduced as a new technical committee in the Organization for the Advancement of Structured Information Standards (OASIS), establishes a single, comprehensive protocol for communication between enterprise key management servers and cryptographic clients. By defining a protocol that can be used by any cryptographic client, ranging from a simple automated electric meter to very complex disk-arrays, KMIP enables enterprise key management servers to communicate via a single protocol to all cryptographic clients supporting that protocol. Through vendor support of KMIP, an enterprise will be able to consolidate key management in a single enterprise key management system, reducing operational and infrastructure costs while strengthening operational controls and governance of security policy.

KMIP therefore represents a major step forward in securing information across the enterprise. It addresses the critical need for a comprehensive key management protocol built into the information infrastructure, so that enterprises can deploy effective unified key management for all their encryption, certificate-based device authentication, digital signature and other cryptographic capabilities.

Introduction

Cryptographic capabilities exist in many places in the enterprise, serving many different purposes. Digital certificates play a critical role in protecting information as it moves within and beyond the enterprise, forming the basis for secure communication. Symmetric key encryption is an essential mechanism for protecting data at rest in such environments as laptops, magnetic tape and disk-arrays. Encryption is also the basis of digital rights management, used to control access to intellectual property such as music and videos. Digital signatures are used to guarantee the authenticity of identities and information. In all these ways, cryptographic capabilities provide a core technology for reducing the risk of unauthorized access or unintentional exposure of sensitive data.

For example, encryption using symmetric keys is an increasingly critical security technology, addressing as it does certain risks and issues that are difficult to address in other ways. For the protection of data in long-term storage, such as on magnetic tape, the cryptographic transformation of data builds security into the data itself, rather than relying solely on physical and technological mechanisms that control access to the data. Access control is still required, both to achieve multiple layers of protection of the data, or defense-in-depth, and to protect the keys used to encrypt the data. But the use of encryption provides a uniquely powerful way to secure information. For this reason, encryption has been identified in a number of regulations as a “safe harbor” technology for protecting sensitive information.

The increasing use of symmetric key encryption, however, brings with it certain dangers. Most important is the risk that the data, once encrypted, cannot be decrypted because the key for that encrypted data has been lost. For this reason, applications, devices and other systems using symmetric key encryption need to be supported by robust key management systems that ensure that keys cannot be lost or misused. Figure 1 shows an example of such a key management system for a tape encryption environment.

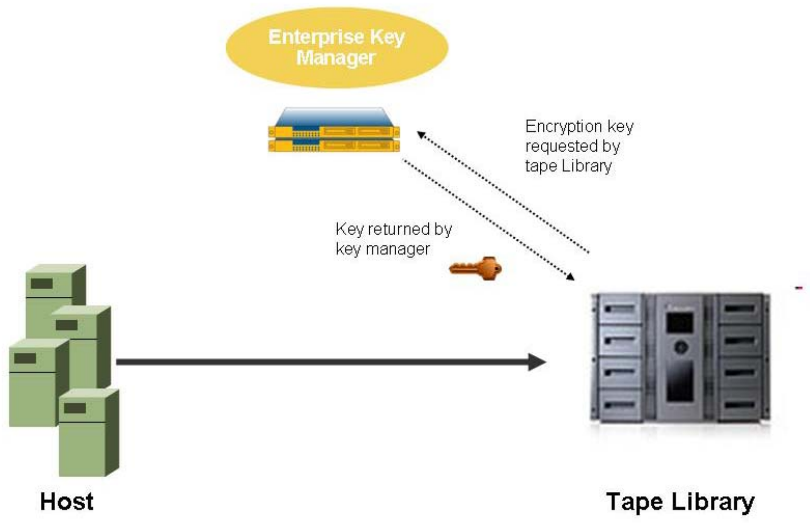
Enterprise Key Manager
Encryption key requested by tape Library
Key returned by key manager
Host
Tape Library

Figure 1: Encryption and Key Management

In this example, the key management server generates keys for the encryption operations performed in a tape library, maintains those keys over long periods of time and controls access to those keys. In this diagram, the tape library acts as a client to the key management server, requesting keys as required for its encryption operations.

However, an enterprise often has to have multiple key management systems, each addressing one or more of the cryptographic client systems in the enterprise. For example, there may be one key management system for tape encryption system, another for managing device authentication certificates, another for managing asymmetric key pairs used in secure communications and so on. This common proliferation of key management systems is shown in Figure 2.

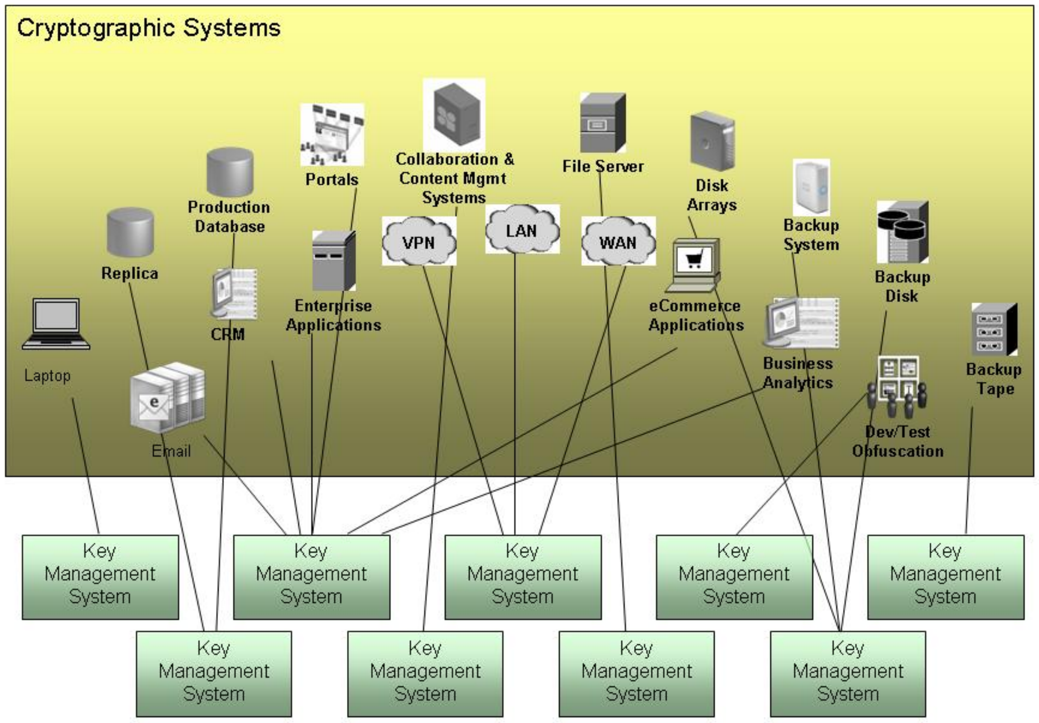

Figure 2: Proliferation of Key Management Systems

Having a unique key management system for each cryptographic client has the advantage that the key management can be tailored to that particular cryptographic environment. This is often the case for laptop encryption, for example, in which the recovery context is typically bound closely with the key used to encrypt a laptop disk.

However, such an approach has significant disadvantages for enterprises deploying multiple encryption systems and other cryptographic clients. It results in increased operational costs, due to the need to maintain expertise in these different key management systems and to perform common operations, such as the definition of key-related security policies, multiple times in multiple key management systems. It also results in increased infrastructure costs, since each vendor supplying a cryptographic client incurs the cost of developing and testing the corresponding key management system; costs that get passed along to the enterprise in higher product and support costs. In addition, the proliferation of key management systems results in higher risk for the enterprise by increasing the likelihood of discrepancies in key-related security policies, the difficulty of oversight for key management processes and the potential failure of key protection processes that could result in loss or misuse of keys.

Each of the cryptographic clients, in communicating with its key manager, typically uses a proprietary format for its messages. This message format, or protocol, usually includes a proprietary definition of the elements of the message, often defining what objects are exchanged between the key manager and a cryptographic client, and what operations are to be performed. These elements are usually assembled in a proprietary way. The protocols typically differ in terms of how the information is secured as it moves across the network, such as whether it uses transport level security, encrypts the message in addition to or instead of using transport security, or applies a digital signature or other authentication to the message in addition to encrypting it.

Enterprise key management systems address the proliferation of key managers by providing a single key management environment that supports multiple cryptographic clients. However, as shown in Figure 3, the many proprietary protocols in use by cryptographic clients mean that every enterprise key management system currently has to support a multitude of different communication mechanisms, often one for each of the different cryptographic clients it supports.

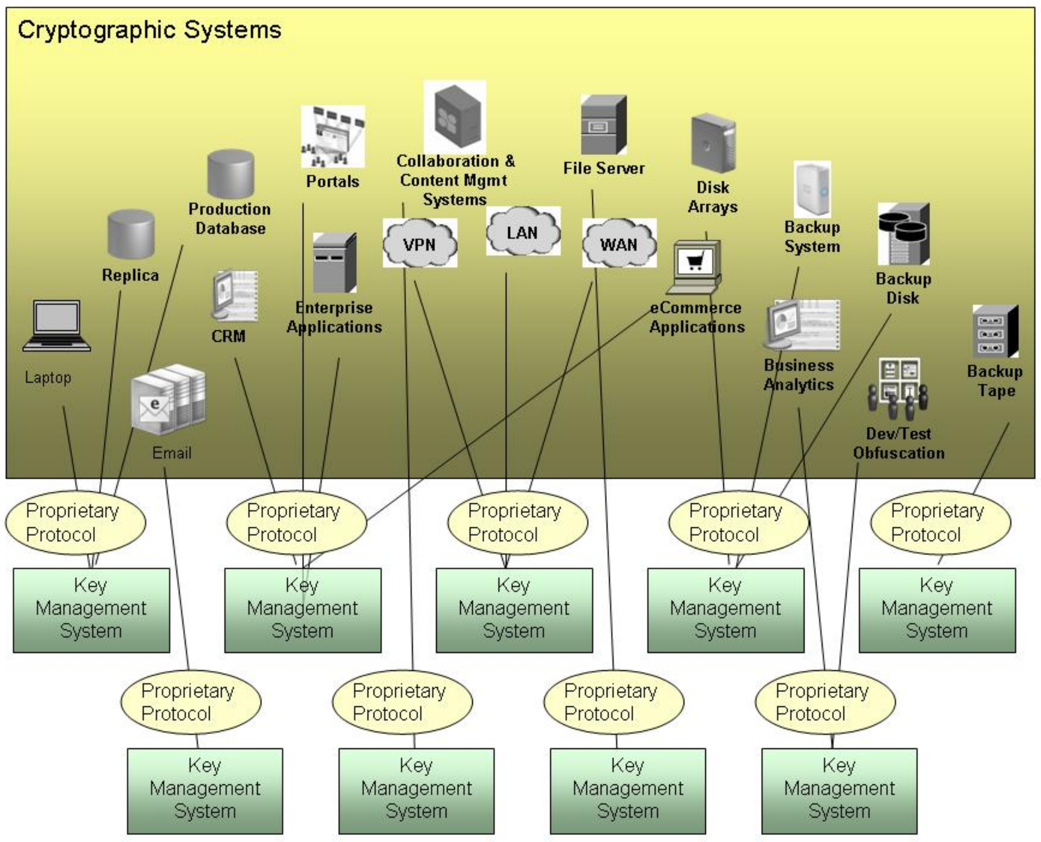

Figure 3: Multiple Protocols for Key Management

These discrepancies between protocols for key management increase the cost of the key management system development and testing, as well as causing delays in having new cryptographic clients supported by a single enterprise key manager. Multiple protocols also increase the risk for enterprises adopting cryptographic capabilities, because of potential differences in how the enterprise key management system supports the systems using these capabilities.

For example, some tape encryption systems that support enterprise key management do so only for the single capability of vaulting their keys into the enterprise key manager for secure, long-term storage. In this model, the keys are generated not in the enterprise key management system but in a key manager local to the tape encryption system. An enterprise that has such a tape encryption system might also have an encryption system for disk-arrays. But this disk-array encryption system might integrate directly with the enterprise key management system for all key-related operations, including defining key policy, generating keys, archiving keys and so on. These different models for the relationship of the key management system and the cryptographic client mean that operations staff will have to have different operational procedures for managing encryption keys for the tape system and for the disk-array system, even though both are supported by the same enterprise key management system.

The Key Management Interoperability Protocol (KMIP) addresses this problem by defining a single format for messages between key management systems and cryptographic clients, as shown in Figure 4.

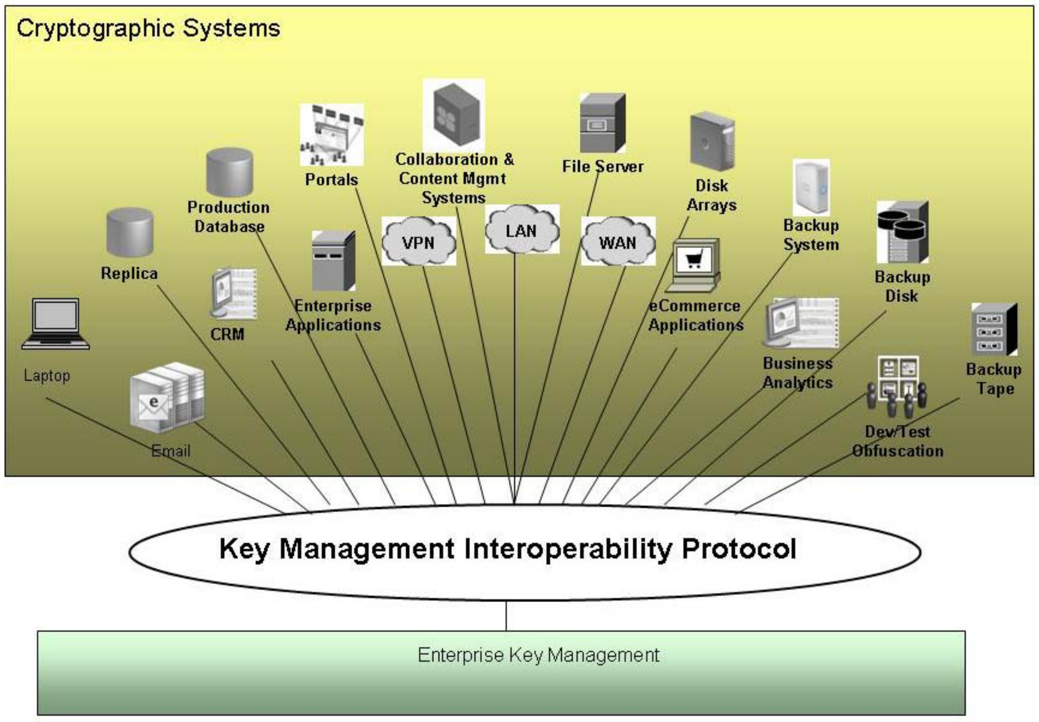

Figure 4: Enterprise Key Management with KMIP

With KMIP, the enterprise key manager can talk to all participating cryptographic clients by means of a single consistent model of objects, operations and attributes. KMIP, therefore, addresses the critical requirement for the simplification of the enterprise key management infrastructure. In doing so, it can help reduce operational costs, reduce key management system costs and reduce the risk in deploying cryptographic capabilities.

The Key Management Interoperability Protocol

The problem addressed by KMIP is primarily that of standardizing communication between cryptographic clients that need to consume keys and the key management systems that create and manage those keys. By defining a low-level protocol that can be used to request and deliver keys between any key manager and any cryptographic client, KMIP enables fully interoperable key management. Through this interoperability, enterprises will be able to deploy a single enterprise key management infrastructure to manage keys for all applications, devices and systems in the enterprise that require symmetric keys, asymmetric keys pairs, digital certificates or other cryptographic objects.

KMIP leverages other standards whenever possible. For example, KMIP uses the key life-cycle specified in NIST special publication 800-57 to define attributes related to key states. KMIP uses network security mechanisms such as SSL/TLS and HTTPS to establish authenticated communication between the key management system and the cryptographic client. KMIP relies on existing standards for encryption algorithms, key derivation and many other aspects of a cryptographic solution, focusing on the unique and critical problem of interoperable messages between key management systems and cryptographic clients.

Defining the Protocol

KMIP includes three primary elements:

- Objects. These are the symmetric keys, asymmetric keys, digital certificates and so on upon which operations are performed.

- Operations. These are the actions taken with respect to the objects, such as getting an object from a key management system, modifying attributes of an object and so on.

- Attributes. These are the properties of the object, such as the kind of object it is, the unique identifier for the object, and so on.

One of the most important features of KMIP is its support for multiple types of cryptographic objects. For example, encryption for data at rest typically uses symmetric keys, including for data encryption on tapes, data encryption for disk-arrays, full-disk-encryption for laptops, file-based encryption and database encryption. Figure 5 shows KMIP being used between a key management system and a broad range of cryptographic clients performing encryption using symmetric keys.

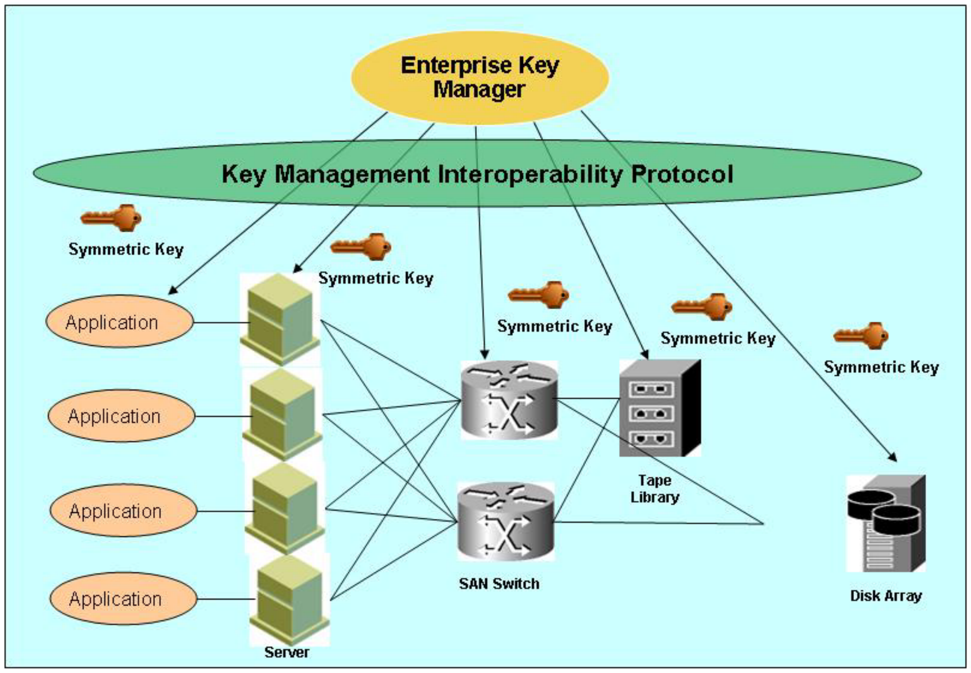

Figure 5: Using KMIP for Symmetric Key Objects

The granularity of the encryption being performed, the algorithms, modes and key strengths being used, and the key lifetimes may be very different across these different encryption environments. Nonetheless, KMIP plays the same essential role for each of these cryptographic clients, enabling their interoperable communication with the enterprise key manager.

Enterprise key management systems are increasingly important for support of other kinds of cryptographic objects, such as X.509 digital certificates used in authenticating applications, devices and systems. As shown in Figure 6, an electric utility could use KMIP in support of renewing the digital certificates used in authenticating automated meters. In such an environment, it is important to be able to preserve the integrity of the usage reports sent from the meter to the utility's billing server. KMIP can play an important role in enabling certificate renewal on a periodic basis for these meters, all of which can communicate with the key management system even though they may have very different capabilities in processing power, memory and connectivity.

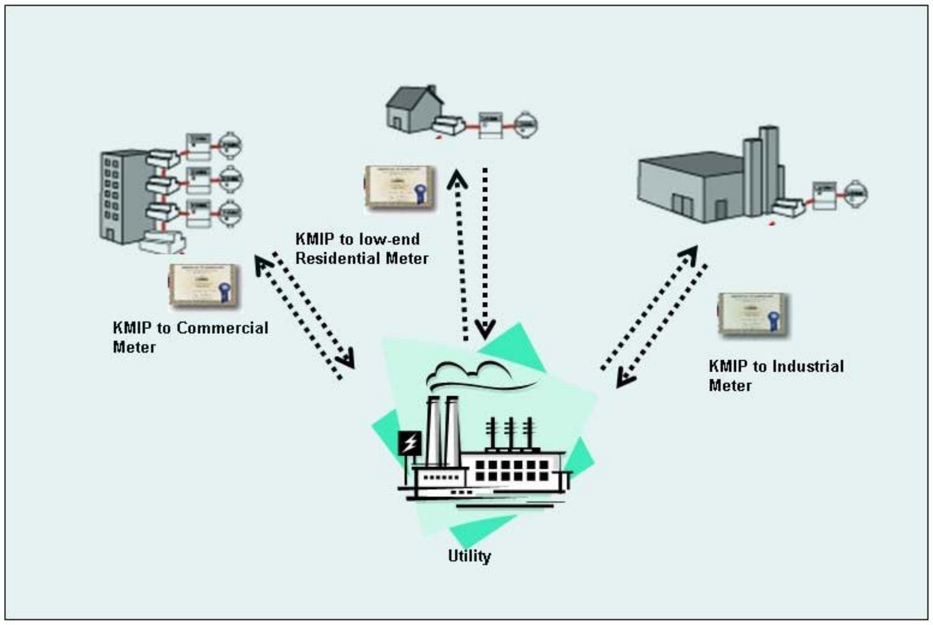

Figure 6: Using KMIP for Digital Certificate Objects

Asymmetric key pairs are also important cryptographic objects in support of enterprise requirements such as message authentication. Figure 7 shows KMIP being used to distribute public keys to partners so that they can be confident that a message, encrypted using the corresponding private key of the key pair, originated from a trusted source.

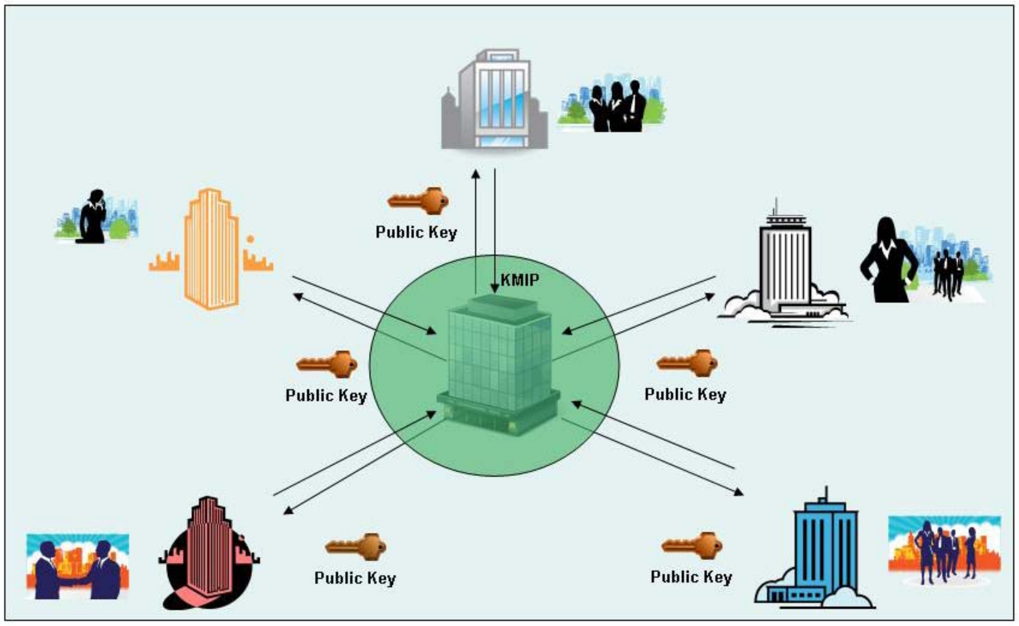

Figure 7: Using KMIP for Asymmetric Key Pair Objects

.

KMIP defines a standard message format for exchanging these and other cryptographic objects between enterprise key managers and cryptographic clients, as shown in the tape encryption example in Figure 8.

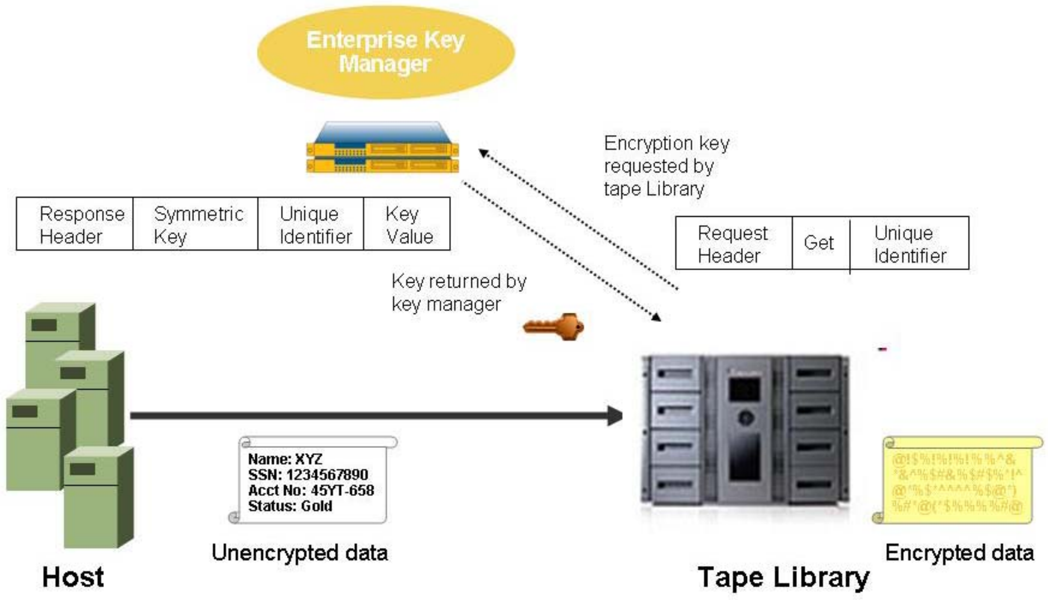

Figure 8: KMIP Request and Response Example

In this diagram, a tape library with encrypting tape drives has received information from a host system in plaintext form and needs to encrypt that information when writing it to tape. The tape system sends a request to the key management system for a “Get” operation, passing the unique identifier for the cryptographic object, in this case a symmetric encryption key that it needs to use to encrypt that particular information. The key management system returns attributes for that object, including not only the value for that key, but also other attributes, such as the kind of key (symmetric) and the unique identifier, that allow the storage system to be sure it is receiving the correct key. Headers for both the request and response provide information such as the protocol version and message identifiers that the participating systems can use to track and correlate the messages.

KMIP also supports including multiple operations within a single message, as shown in Figure 9.

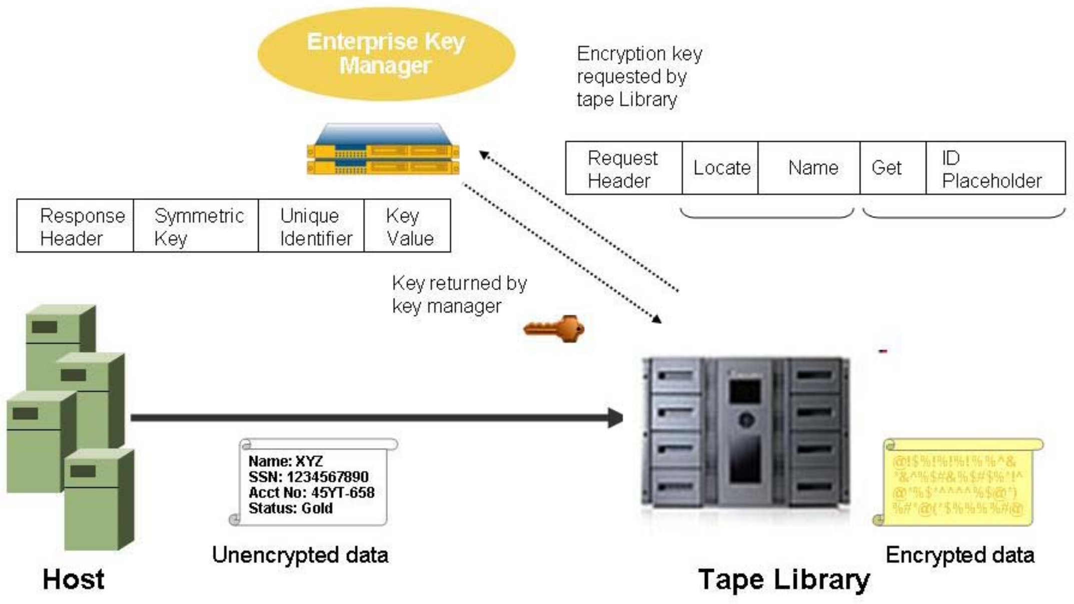

Figure 9: Supporting Multiple Operations in a KMIP Message

In this example, the tape system requests the key management server to use a "locate" operation to find a key based on a "name" attribute. Once the server has located the key, it then uses the unique identifier attribute for that key, indicated in the request message by the "id placeholder" attribute, to retrieve the key, assemble a response message and return the response to the tape system.

KMIP request and responses messages are constructed by assembling the message in a tag/type/length/value format, as shown in Figure 10.

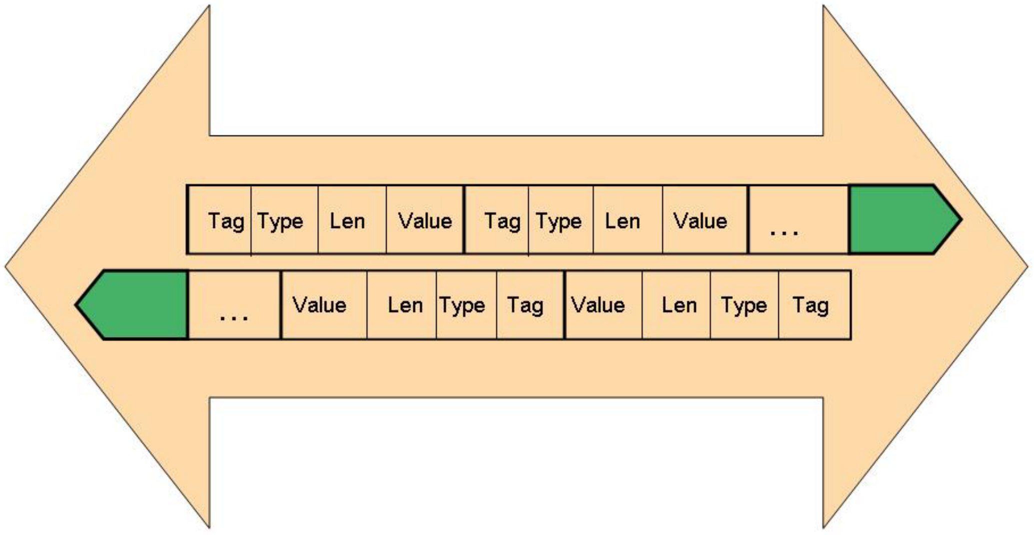

Figure 10: KMIP Message Representation in TTLV Format

For example, in the request in the tape encryption example above (see Figure 8), the message could include the following representation for the "get" operation and the "unique identifier" attribute.

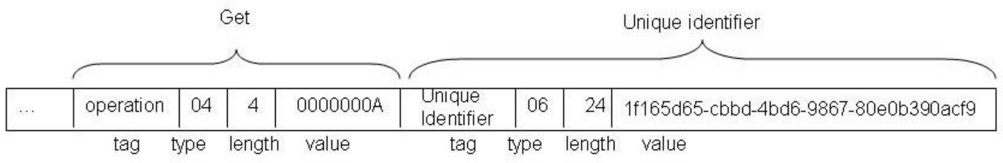

Figure 11: KMIP Get Message Representation

The protocol supports other elements, such as the use of templates that can simplify the specification of attributes in a request or response. But at its most basic level, KMIP consists of placing objects, operations and/or attributes either into a request from a cryptographic client to a key management server or into a response from a key management server to a cryptographic client.

KMIP is not an application programming interface; that is, KMIP does not define a set of services called by an application. Nor is it an object model with corresponding methods called by an application. Rather, it specifies a protocol, the format in which a message is constructed and the elements that are included in the message. This allows KMIP to be used by any cryptographic client, from the very smallest devices to the most complex storage arrays, that needs to get security objects. Expressing KMIP as a wire protocol, as shown in Figure 12, is particularly valuable in order to support those encryption environments with limited processing power or network bandwidth.

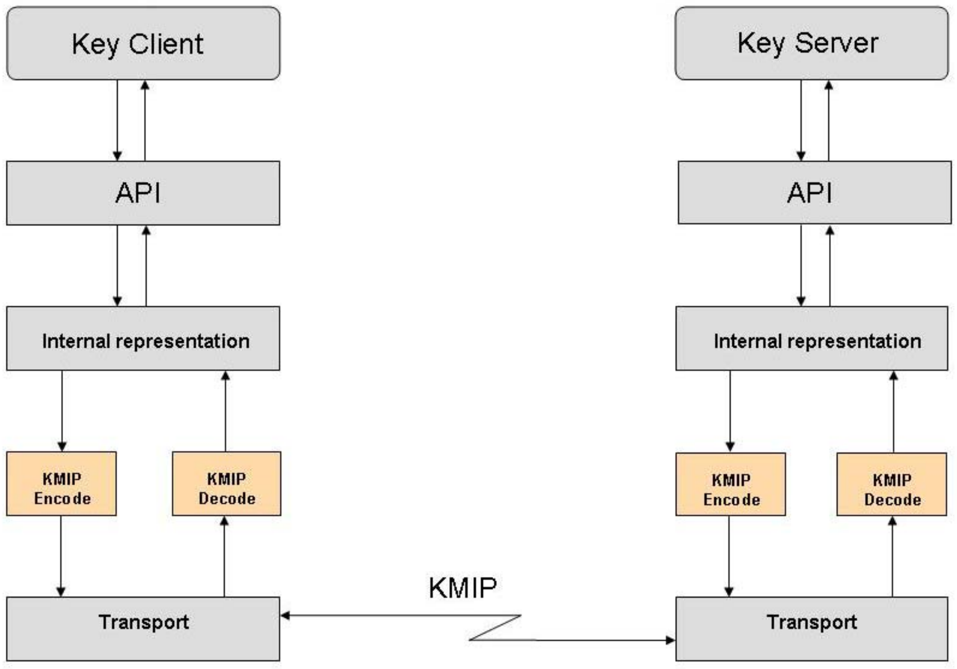

Figure 12: KMIP as Wire Protocol

By specifying the message contents at this level, KMIP ensures that the recipients of the messages know precisely what operation is requested or has been performed, for which object, with what attributes. This detailed understanding of the message is essential to ensure interoperability across different implementations of key management systems and cryptographic clients.

In KMIP, request messages can originate either in a cryptographic client or in a key management system. KMIP is primarily designed for client-initiated requests; that is, requests should be initiated by the cryptographic client with a corresponding response from the key management system, such as a request to return a key (get), a request to find a key (locate) and so on. However, KMIP also supports a small number of operations that can be initiated by the key management system, such as a notification to a particular cryptographic client that an encryption key currently in use should be replaced by a new encryption key for subsequent encryption operations.

Elements of KMIP

KMIP includes security objects, operations for the objects, and attributes that can be associated with those objects. Each of these areas is discussed in the sections that follow.

KMIP Objects

KMIP is designed to support all security objects that need to be distributed between a key management server and a cryptographic client.

Table 1: KMIP Objects

Object	Definition
Certificate	A digital certificate, such as an X.509 certificate.
Opaque Object	An object stored by a key management server but not necessarily interpreted by it.
Policy Template	A stored, named list of policy-related attributes.
Private Key	The private portion of an asymmetric key pair.
Public Key	The public portion of an asymmetric key pair.
Secret Data	A shared secret that is not a key or certificate.
Split Key	A secret, usually a symmetric key or a private key, which is split into a number of parts, which can then be distributed to several key holders, for additional security.
Symmetric Key	A symmetric encryption key or message authentication code (MAC) key.
Template	A stored, named list of KMIP attributes.

In most key management systems currently in use, security objects tend to be symmetric keys used for data encryption. For example, a storage environment in which tape encryption is performed at the tape drive typically uses symmetric keys as input to the algorithm for the encrypted data being written to tape. Similarly, within a laptop environment using full-disk encryption, that encryption is usually performed with symmetric key encryption algorithms.

Symmetric keys are in widespread use because of the speed of symmetric key operations. For those situations where a large number of encryption operations need to be performed and where encryption speed is critical, symmetric keys provide considerable performance advantages. However, they have the disadvantage that any process that has possession of that key is able to decrypt the information. Therefore, the security of information encrypted with symmetric keys may be at risk if, for example, a large number of systems are given the same symmetric key to decrypt shared information.

The risk of exposure for the security object is often addressed by using asymmetric key pairs rather than symmetric encryption keys. With asymmetric key pairs, one key is used to encrypt information and a second key, mathematically related to the first key but not identical, is used to decrypt the information. Asymmetric keys are therefore particularly important for digital signatures and entity verification. Managing asymmetric key pairs has been increasingly critical for enterprise key management systems, particularly in terms of device certificates, SSL certificates and other entity identification purposes. Key management systems handle distribution of certificates, as well as life-cycle operations such as renewal, expiration and revocation.

KMIP also supports other kinds of security objects by allowing “opaque objects” whose properties are not directly visible in the protocol. Opaque objects allow a key management server to manage objects that it could not otherwise. For example, a key management server could store, as an opaque object, a key for an algorithm that the server does not otherwise support. Or a key management server could store, as an opaque object, a wrapped key that the server cannot unwrap.

Templates are a special kind of KMIP object used to simplify the specification of attributes for KMIP objects. A template is an arbitrary grouping of attributes, mutually understood by the cryptographic client and the key management system, that can be passed in request and response messages. For example, a cryptographic client can specify a template when requesting a new key, identifying the attributes that it would like the key management system to assign to the created key. Templates are not defined by KMIP and must be coordinated between the encryption and key management systems to ensure consistency between the two environments.

KMIP Operations

When an encryption environment sends a request to a key management server, it indicates not only the object in which it is interested, but also the operation it wants the key management server to perform on that object. For example, in data encryption environments such as those related to storage, that request will often relate to the generation of a new key or the retrieval of an existing key. It may also request information about a key, or modify that information.

The kinds of operations that can be performed vary somewhat depending on the kind of security object, particularly for certificates compared to other kinds of security objects. They also vary depending on whether the operation is initiated by the cryptographic client or by the key management system. Table 2 below shows the object-related operations that are initiated by the cryptographic client (except for the certificate-specific operations).

Table 2: KMIP Operations

Operation	Definition
Activate	Requests the key management system to activate an object.
Add Attribute	Requests the key management system to add a new attribute to an object and set the attribute value.
Archive	Requests that an object be placed in archival storage by the key management system.
Check	Requests that the key management system checks for the use of an object according to specified attributes.
Create	Requests the key management system to generate a new key.
Create Key Pair	Requests the key management system to generate and register a new public/private key pair.
Delete Attribute	Requests the key management system to delete an attribute for an object.
Derive Key	Request that the key management system derive a symmetric key using a key or secret that is already known to the key management system.
Destroy	Indicates to the key management system that the key material for the object should be destroyed.
Get	Requests that the key management system returns an object, which is specified in the request by its Unique Identifier attribute.
Get Attributes	Requests one or more attributes of an object.
Get Attribute List	Requests a list of the attribute names associated with an object.
Get Usage Allocation	Requests an allocation from the current Usage Limits values for an object, to allow the client to use the object for protection purposes.
Locate	Requests that the key management system searches for one or more objects, specified by one or more attributes.
Modify Attribute	Requests the key management system to modify the value of an existing attribute.
Obtain Lease	Requests a new Lease Time for a specified object.
Query	Interrogates the key management system to determine its capabilities and/or protocol mechanisms.
Recover	Requests access to an object that has been placed in archival storage (via the Archive operation).
Register	Requests the key management system to register an object passed to it in this operation.
Re-key	Requests the key management system to generate a replacement key for an existing symmetric key. It is analogous to the Create operation, except that many of the attributes of the new key are unchanged from the original key.
Revoke	Requests the key management system to revoke an object.

For certificates, many of the operations are comparable to those performed for other security objects. However, a key management server is typically not the certification authority (CA), but serves as a proxy to a CA. This relationship between the cryptographic client, the key management server and the certification authority is shown in Figure 13.

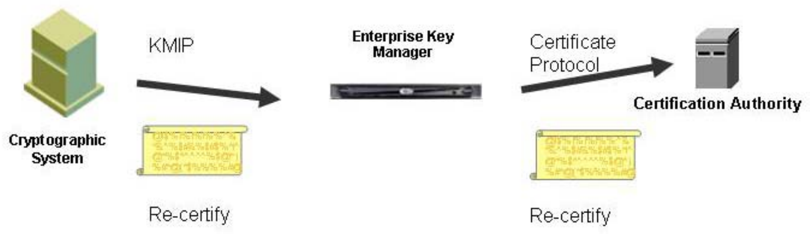

Figure 13: KMIP Support for Certificate-related Operations

Some operations supported by certificate management protocols, such as the XML Key Management Specification (see http://www.w3.org/TR/xkms2), are not specified in KMIP. Rather, KMIP supports those elements of certificate management that are typically handled by a key management system, as a proxy for a certification authority, for infrastructure certificates such as device certificates and SSL certificates. The certificate-specific operations in KMIP are shown in the table below.

Table 3: KMIP Certificate-specific Operations

Operation	**Definition**
Certify	Requests a new certificate for a public key or renewal of an existing certificate with a new key.
Re-certify	Requests the renewal of an existing certificate with the same key pair.
Validate	Requests the validation of a certificate chain, and return of information on its validity.

KMIP supports both synchronous and asynchronous request/response models in the protocol. Asynchronous requests return a pending status to the requester. The requester then uses a “poll” operation to check on the status of the outstanding asynchronous operation. The requester can also issue a “cancel” operation to end pending operations, as shown in Table 4 below.

Table 4: Asynchronous Operations

Operation	**Definition**
Cancel	Used to cancel an outstanding asynchronous operation.
Poll	Used to poll the server in order to obtain the status of an outstanding asynchronous operation.

In general, KMIP requests are sent from a cryptographic client to a key management system. However, for certain special circumstances in which the key management system knows how to contact a particular cryptographic client, the key management system can send a request message, initiating the request/response sequence with that cryptographic client. KMIP supports two operations for this situation, notify and put, shown in Table 5 below.

Table 5: Server-Initiated Operations

Operation	Definition
Notify	Used to notify a client of events.
Put	Used to "push" Managed Cryptographic Objects to clients.

KMIP Attributes

KMIP supports a broad range of attributes for security objects, with the attributes either sent from the cryptographic client to the server, such as sending the unique identifier on a "get" operation so that a particular key can be retrieved, or returned from the key management system to the cryptographic client. Attributes vary depending on the security object, with a core set of attributes that are specified for all objects (such as state), complemented by object-specific attributes when needed (such as the certificate issuer and other certificate-specific attributes).

Table 6 shows the attributes defined in KMIP for cryptographic objects.

Table 6: KMIP Attributes

Operation	Definition
Activation Time	The date and time when the object may begin to be used.
Application Specific Identification	The intended use of a Managed Object.
Archive Date	The date and time when the object was placed in archival storage.
Certificate Issuer	An identification of a certificate, containing the Issuer Distinguished Name and the Certificate Serial Number.
Certificate Subject	The subject of a certificate, containing the Subject Distinguished Name.
Certificate Type	The type of a certificate, such as X.509 or PGP.
Compromise Occurrence Date	The date and time when an object was first believed to be compromised.
Compromise Date	The date and time when the object is entered into the compromised state.
Contact Information	The name of an entity to contact regarding state changes or other operations for the object.
Cryptographic Algorithm	The cryptographic algorithm used by the object, such as RSA, DSA, DES, 3DES, or AES.
Cryptographic Length	The length in bits of the cryptographic key material of the

	object.
Cryptographic Parameters	A set of optional fields that describe certain cryptographic parameters to be used when performing cryptographic operations using the object, such as hashing algorithm.
Cryptographic Usage Mask	A bit mask that defines which cryptographic functions may be performed using the key.
Custom Attribute	User-defined attribute intended for vendor-specific purposes.
Deactivation Date	The date and time when the object may no longer be used for any purpose, except for special circumstances requiring decryption, signature verification, or unwrapping,
Destroy Date	The date and time when the object was destroyed.
Digest	A digest of the key (digest of the Key Material), certificate (digest of the Certificate Value), or opaque object (digest of the Opaque Data Value).
Initial Date	The date and time when the object was first created or registered at the key management system.
Last Changed Date	The date and time of the last change to the contents or attributes of the specified object.
Lease Time	The time interval during which a client should use the object.
Link	A link from an object to another, closely related object, such as the original object which has been replaced in a re-key operation by the object for which the link attribute is defined.
Name	A descriptive name for the object, assigned by the cryptographic client to identify and locate an object
Object Group	The name of a group to which the object belongs.
Object Type	The type of object, such as public key, private key, or symmetric key.
Operation Policy Name	An indication of what entities may perform which key management operations on the object.
Owner	The name of the entity that is responsible for creating the object.
Process Start Date	The date and time when an object may begin to be used for process purposes, such as decryption or unwrapping.
Protect Stop Date	The date and time when an object may no longer be used for protect purposes, such as encryption or wrapping,
Revocation Reason	An indication of why the object was revoked, such as “compromised”, “expired” or “no longer used”.
State	The state of an object as known to the key management system.
Unique Identifier	A value generated by the key management system to uniquely identify an object.
Usage Limits	A mechanism for limiting the usage of an object, such as to no more than a specified number of bytes that can be encrypted with a particularly symmetric key.

One of the attributes specifiable in KMIP is the optional “name” attribute, which can have multiple values for a single object. KMIP does not specify a namespace for objects and attributes, supporting instead a number of formats for how names can be expressed, including formats such as URI-based names (in support of globally unique names). Global uniqueness of keys is particularly important for certain environments, particularly for tape cryptographic clients in which keys are long-lived. In such environments, the specification of a globally unique name, if the name attribute is used, may be important because of the possibility of merging of key management systems over time. In such a case, a name that had been unique in a one particular key management environment may collide with a name assigned in another key management environment unless care is taken to differentiate those namespaces. Use of a URI as a naming convention can help to minimize the risk of name collision, as shown in Figure 14.

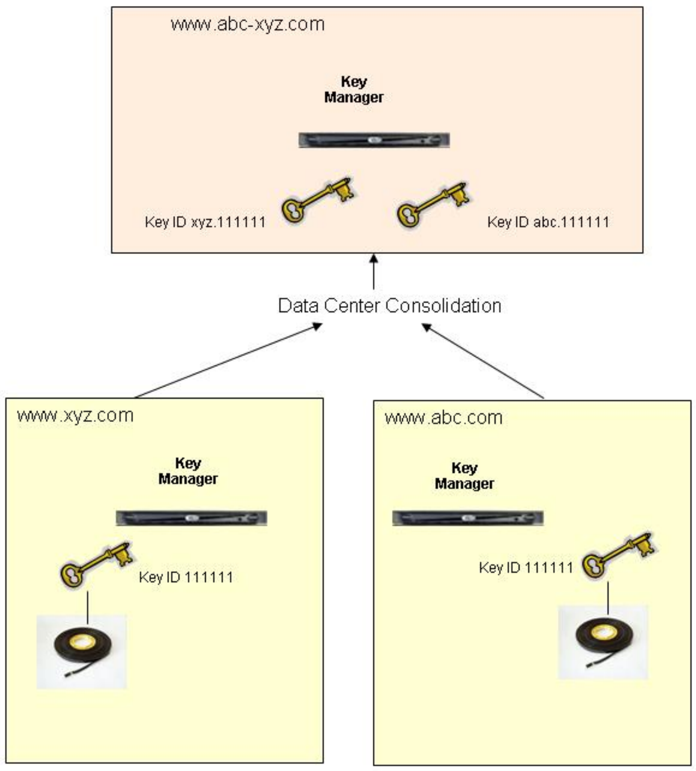

Figure 14: Name Attribute

The state attribute defines the uses for a key. It follows the state definitions in NIST 800-57, shown in Figure 15.

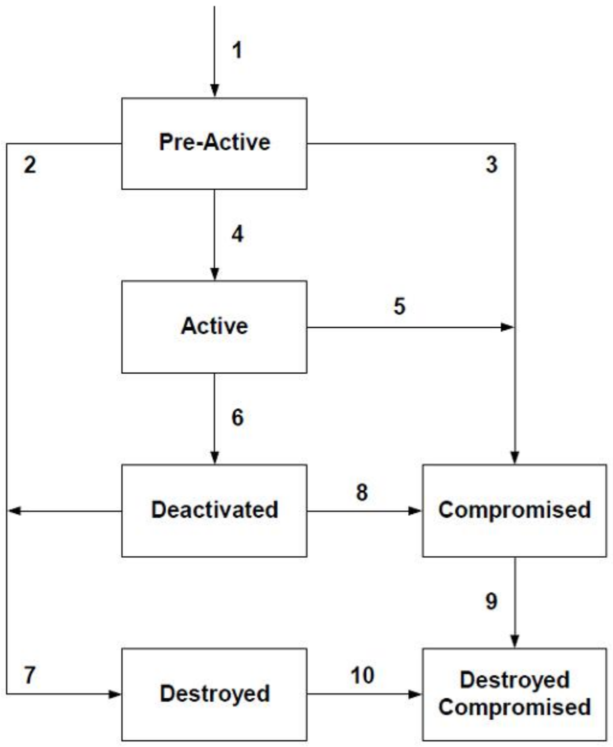

Figure 15: States and Transitions

An object (i.e., key) moves from one state to another as a result of operations performed on the object, rather than as a result of changing the state attribute directly. The state attribute, therefore, describes the result of transitions rather than being used as a settable value that forces the transition of an object. The time-related attributes reflect these changes of state, from the initial creation time for the object through its destruction.

When a re-key or re-certify operation is requested on a security object, the time-related attributes assigned for the re-keyed object could become ambiguous. Therefore KMIP specifies the expected behavior with regard to setting those time-related attributes. For example, when a re-key is performed, the initial time for the new key should be set to the time of the new object being created. Related time attributes (such as activation time) should be established as an offset of the activation time, based on the intervals of the original object. Though the KMIP protocol does not enforce these relationships, consistent use of the time attributes improves interoperability across key management systems by ensuring consistent behavior for objects.

In addition to name, state, and time-related attributes, KMIP also supports a number of cryptographic attributes for objects, including algorithm type and other information. These attributes are intended to provide a consistent way of specifying this information so that key-related policies expressed in the key management system can be understood and respected by the cryptographic clients. KMIP cannot enforce behavior in the key management system and the cryptographic client (including the use of attributes). However, the specification of these attributes in the protocol helps to ensure that fine-grained policies regarding security objects can be understood by the cryptographic clients and applied consistently and appropriately.

Authenticating KMIP Servers and Clients

KMIP does not specify, as part of the protocol, the mechanisms by which key management systems and cryptographic clients identify themselves to each other. Rather, KMIP relies on existing standards for mutual authentication that specify how this identification is to be established. KMIP currently defines two authentication profiles, the first based on TLS, the second on HTTPS. In both profiles, digital certificates are used by the client and the server to identify themselves as participants in KMIP requests and responses, as shown in Figure 16.

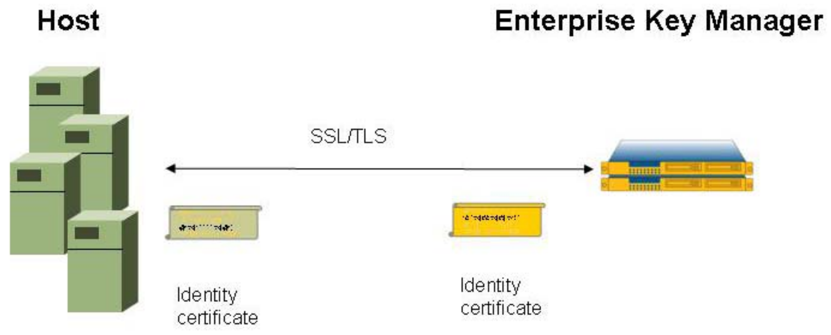

Figure 16: Mutual Authentication

Registration mechanisms by which the enterprise key manager learns the identity of cryptographic clients are not defined in KMIP. The credentials used by the cryptographic client to identify itself can be included in the protocol as part of a request message, to simplify processing of the request by the key management system. However, the credential element is not guaranteed to be authenticated and is therefore not intended for use in authentication.

Message integrity for KMIP exchanges, as well as entity authentication, is provided by TLS. Other mechanisms that could also be used for enhanced security of KMIP messages are not currently defined for KMIP.

Conclusion

KMIP represents a significant step forward in securing information infrastructure throughout the industry. KMIP's creation and subsequent adoption by industry vendors will reduce the complexity of encryption management for enterprises by building interoperability into the key management environment. By enabling support for interoperability between cryptographic clients and enterprise key management systems, KMIP reduces infrastructure costs and the risks in adopting cryptographic solutions as an essential element of securing information, identities and infrastructure.