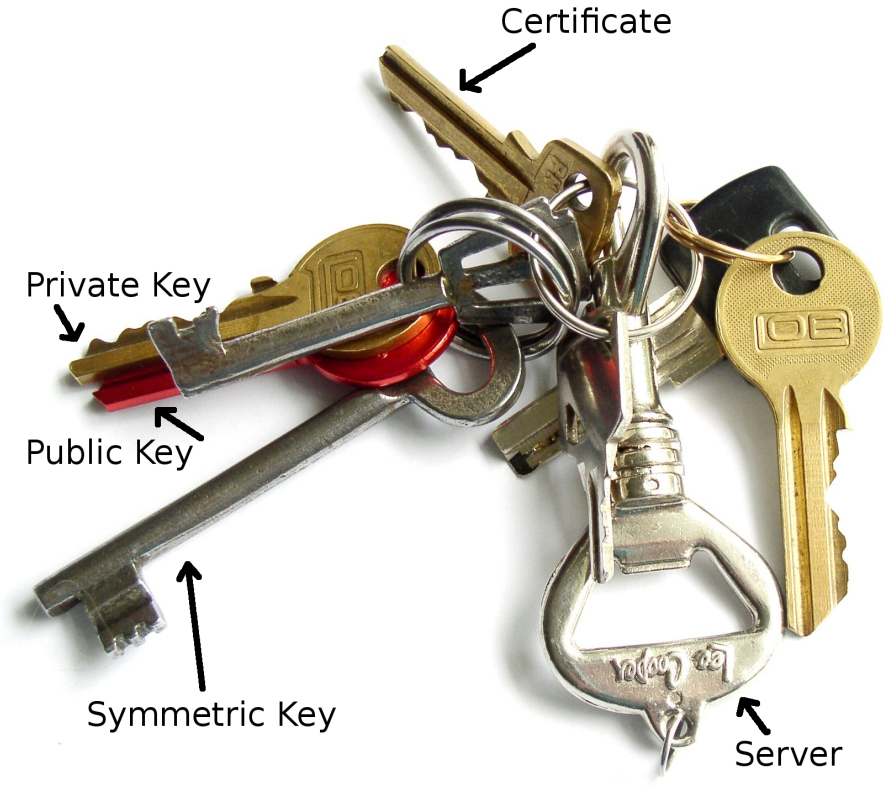
UNIVERSITY OF AMSTERDAM

System and Network Engineering
Large Installation Administration Project
Master of Science Program

Academic year 2008–2009




# *Cryptographic Key Management*

by

Sevickson Kwidama
sevickson.kwidama ⇒ os3.nl
Taarik Hassanmahomed
taarik.hassanmahomed ⇒ os3.nl

Large Installation Administration report for System and Network Engineering, University of Amsterdam, the Netherlands, for academic year 2008–2009.


Sevickson Kwidama | `sevickson.kwidama` ⇒ `os3.nl`.
Taarik Hassanmahomed | `taarik.hassanmahomed` ⇒ `os3.nl`.



Cover image: This image symbolizes the different cryptographic keys used in a server environment. Source: `http://www.sxc.hu/photo/24847`, added text and arrows to the image.

Version 1.0.0 and compiled with LaTeX on March 29, 2009.

# Contents

# List of Tables



# List of Figures

# 1 Introduction

This document is the culminating work of a project for the course LIA (Large Installation Administration), in the study SNE (System and Network Engineering) at the UvA (Universiteit van Amsterdam).

The title of this report is 'Cryptographic Key Management'.
With this title we mean the management of keys used for encryption, decryption or verification in an organisation's infrastructure.
We came to the choice for this subject because of different news articles we came across, about current standardisation efforts and open-source implementations of key management systems.

Cryptography is used nowadays in a diverse range of systems, infrastructures and devices. These include encrypting file systems, databases, encryption-enabled devices like disks and tape drives. When using cryptography there comes a need to maintain the keys used. Key management is thereby driven by the tools and applications where it is used.
What you get is that system administrators manage keys used for different operating systems, database administrators manage keys used for database field-level encryption, storage administrators manage keys used for storage device-level encryption, etc.[3]

To avoid the problem of different administrators with their own keys we will be looking at centrally managing keys in an organisation.
In the work proposal for this project we gave a couple of questions we investigated:

1. What is cryptographic key management?
2. Problems and solutions for key management centralisation and administration.
3. Explain various key management standards and implementations.

We will try to answer these questions in this document.
This document is divided in seven sections:

**Cryptographic Key Management** This section gives an introduction to the key management problem and explains the basic idea of centrally managing cryptographic keys.

**Administration** In this section we will be discussing the pros and cons of centrally managing keys, in LIA terms.

**Key Management Lifecycle** The theoretical lifecycle of cryptographic keys passing through a management system are explained.

**Infrastructure** A concept model of a management infrastructure is given.

**Standards** In depth look at standardisation efforts for a centralised cryptographic key management.

**Implementations** Current implementations of key management are explained, specifically looking at key management of backup storage devices.

**Conclusion** The different conclusions that were taken are given.

# 2 Cryptographic Key Management

Like we said in the Introduction, cryptography is used nowadays in a diverse range of systems, which makes the management of the keys even more important, because without the keys cryptography is useless.
This creates the problem of different administrators that manage their own keys in an organisation.
When you have different administrators managing their own keys, they start to become independent from each other, creating different key management "silo's" (sections), as seen in figure 1.

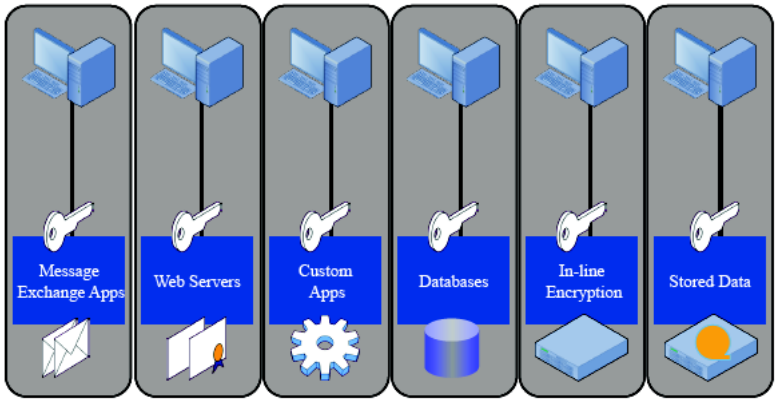



Figure 1: Multiple silo's with different key administrators. Source: [1]

This problem also elevates the risk of key exposure or mishandling of keys, the reason for this is that in most cases there are no common policies and procedures in place to ensure proper and controlled handling of key material over all the systems.

To solve this and other problems a centralised cryptographic key management administration is needed, this does not mean a single key manager administrator with access to every piece of information, but actually a key management system that allows for different people to follow a set of tasks and workflows to manage keys across an organisation in an orderly and consistent manner, that is audited, repeatable and controlled.

> Cryptographic key management is the complete set of operations necessary to nurture and sustain encrypted data and its associated keys during the key and data lifecycles. A key management system is an implementation of all or parts of these key management operations.[4]

The key management operations in the quote above are operations translated from an organisation's security requirements through key management policies.
Key management is more than just encrypting and decrypting data, it is more about managing keys and how those keys are used and audited.
Many applications like PKI (Public Key Infrastructure), HSM (Hierarchical Storage Manager), HSM (Hardware Security Module), SSS (Single Sign-on System) use or provide some kind of key management service, but these applications are not key managers on their own.

Key management consists of management operations on the key and policy operation that govern it's use.
Operations like backup, restoration, archival, retention, expiration, deletion and destruction of key material are part of the management operations and an important part of the key lifecycle.

Other management operations include distribution of key material to multiple locations for performance or reliability, auditing of a given key's lifecycle, reporting of events and alerts.
Policy operations set and manage key policies, which stipulate the use of a specific key and can be as simple as only for decrypting or complex and compelled by rules or laws to use them in a certain way. Other policy operations include generating notifications when policy violations occur, enforcing policies, reporting policy enforcement when rules change to show the transition from the old rules to the new rules. These policies have a key role in the audit part of key management, which securely records all operations associated with keys under it's control and is therefore important when trying to find out what happened when problems occur.

In centralised cryptographic key management, an instance of a provider of key management services is called the KM (Key Management) Server and the consumer the KM Client.
In this client-server model you have a cryptographic module using the keys provided by the KM Client when needed. Whenever the KM Client needs a key, it requests the key from the server through key management services via the client API or optional libraries and receives the requested key from the server.
The most important part of this operation is the standardisation of the key management services for better interoperability and migration between different vendors.[5]

# 3 Administration

Management of cryptographic keys is often a subject that not much attention is paid to. A reason for this is that it is a difficult matter when trying to manage different keys for different entities in a large organisation.
Like it was explained in chapter 2 on page 5, there are different sections, each depicting a part of an organisation that is managing it's own keys.

To solve the problem of different sections, centralisation of the different key management sections is needed. By automating the key generation, distribution and destruction the management can be easier centralised.
Policies can be implemented to maintain the individual cryptographic needs of different departments in an organisation.
The centralisation of the keys also gives a better chance of auditing and reporting the key use.

After centralisation an issue can arise with availability. This issue is amplified when all the keys and their attributes is concentrated on one key server.
You can mitigate this problem by backing up and copying the material to an independent entity and the possibility of a luke-warm standby key server that can access the independent entity at hand.

Key management is at the moment a vendor-locked "feature", this means that if you want to scale your infrastructure you will need to acquire parts from the same vendor.
At the moment of writing there are different organisations that are trying to standardise the interoperability protocol for key management to make it possible for different vendor hardware to work together. The standardisation efforts are explained in chapter 6 on page 12.
These efforts are at the moment in pre-mature phase, meaning that it will take a few years before there are crystallized standards.

Another important part of key management is offcourse security.
After you've consolidated the different keys in a key management infrastructure you are on your way to a secure infrastructure. By adding the different keys in one system you minimise the exposure possibilities of the different keys of your systems.
Now you will have to concentrate on hardening the key management servers both physically and virtually.

# 4 Key Management Lifecycle

To employ a correct key management system, you will need a lifecycle to stipulate when and what state a key can reside in.
The key management lifecycle that we will be discussing in this section is based on the lifecycle given by NIST (National Institute of Standards and Technology) in Special Publication 800-57.[6] The lifecycle of NIST is the basis for different key management lifecycles used nowadays, that is why we will be discussing the NIST lifecycle.

The lifecycle is divided in, management phases and key states/transitions.
There are six key states connected through ten transitions divided over four management phases. In figure 2 the key management lifecycle is displayed.

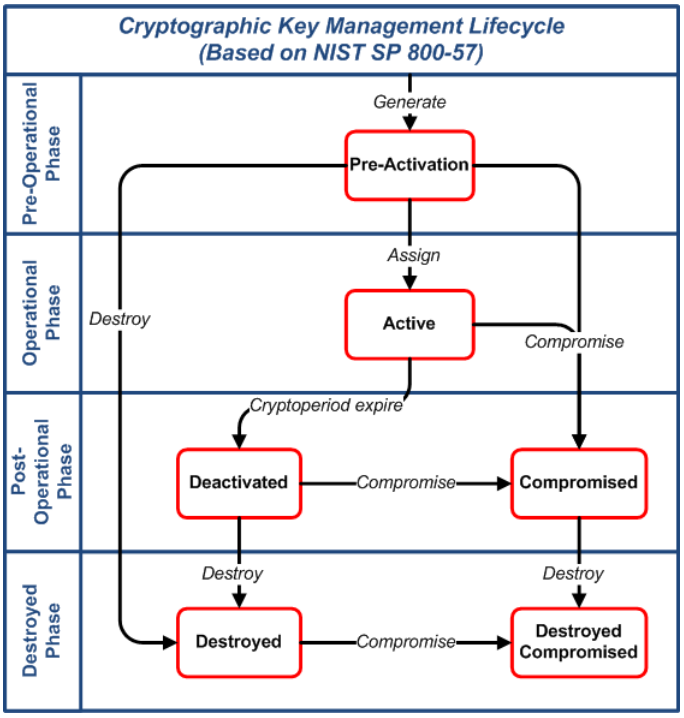



Figure 2: Cryptographic Key Management States and Phases

## 4.1 Management Phases

There are four management phases:

**Pre-Operational** In this phase the keying material is not available for cryptographic operations. Keying material is the data (e.g., keys and IVs) necessary to establish and maintain cryptographic keying relationships.
This means that the keys are not yet generated or the keys are in pre-activation state (see section 4.2).[6]

**Operational** The keying material is in active mode. The keys can be designated as *Protect only*, *Process only* or *Protect and Process*.
Protect only is used for encryption, an example of protect only is a private signature generation key.
Process only is used for decryption, an example is a public signature verification key.

An example of Protect and Process is a symmetric data encryption key, this key can be used for a predefined time period and after that period expires transitions to process only.

**Post-Operational** This phase, the keying material is no longer in normal use. The keys can be used to process information in certain circumstances.
The keys can be in the deactivated or compromised state, in these states the keys are archived when they are not processing information.

**Destroyed** Keys are deleted, they can be in destroyed or destroyed compromised state. The attributes of the keys may be kept for administration or auditing purposes.

## 4.2 Key States

Now that the management phases of the cryptographic key management lifecycle have been explained, we will be turning our attention to the states which the keys can reside in when in this lifecycle.
The keys can traverse any of the states during the lifecycle.

**Pre-Activation State** In the pre-activation state the key was already generated, but not yet in use. While the key is in this state it can be submitted to a CA (Certification Authority) for certication and registration. Or the key can be used to perform key confirmation between different parties.

**Active State** This is the state where the key is active and can be used to encrypt or decrypt data. Encrypt or decrypt data can also be called, protect information for encryption and process previously protected information for decryption.
Protect only is used for encryption, an example of protect only is a private signature generation key.
Process only is used for decryption, an example is a public signature verification key.
An example of protect and process is a symmetric data encryption key, this key can be used for a predefined time period and after that period expired transitions to process only.

**Deactivated State** This state is the state that a key resides in when the cryptoperiod has passed. In this state the key can still be used to process cryptographically protected information.
The key stays in this state until it is not required anymore to process information, after this the key is destroyed.

**Compromised State** When a key is determined by or leaked to an unauthorised entity the key is moved to the compromised state. In this state the key might be used to process information under controlled conditions.

**Destroyed State** The key is destroyed. The key attributes, like name, type and cryptoperiod, might be kept for auditing purposes.

**Destroyed Compromised State** The key is also destroyed in this state and the key attributes might be retained. The only difference with the state above is that in this state the key is known or suspected to have been compromised.

# 5 Infrastructure

The infrastructure in figure 3 is to give a conceptual idea of how a key management infrastructure can look like.

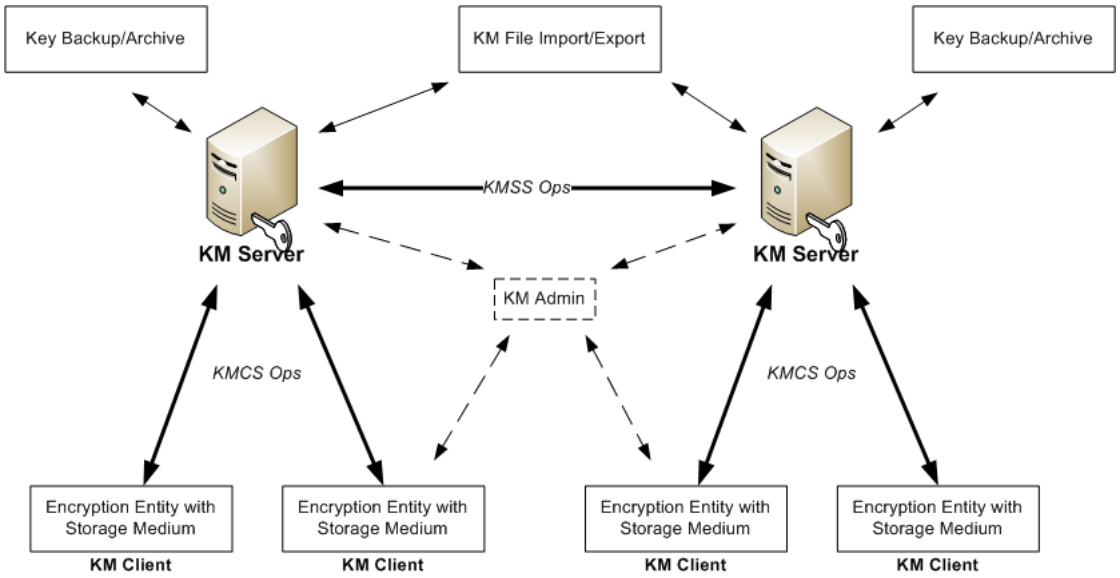



Figure 3: Cryptographic Key Management Infrastructure. Based on [2]

There are different components when you look at the infrastructure, the components are:

- Key Backup/Archive
- KM File Import/Export
- KM Server
- KMSS Ops (Key Management Server-Server Operations)
- KM Admin
- KMCS Ops (Key Management Client-Server Operations)
- KM Client (Encryption Entity with Storage Medium)

The basic idea of this infrastructure is that there is a *KM Server* that provides keying material to *KM Clients*, this is done via *KMCS Ops*.
The other components support these operations by providing backup and archiving of keys (*Key Backup/Archive*) or Server-to-Server communication (*KMSS Ops*) and exchange (*KM File Import/Export*). And all this is managed by a *KM Admin*.

Now that the basic idea has been discussed we will be taking a closer look at each component individually.

**Key Backup/Archive**.
This component can be a software- or hardware-based solution that can provide secure, long-term storage for the keys and their attributes.
The keys currently used can be copied to this component to provide secure key backup for possible disaster recovery. Another option for this component is the archiving of deactivated keys and archiving of the attributes of destroyed keys for auditing purposes.

**KM File Import/Export**.
With this independent component file exchange services is possible between KM Servers.

**KM Server**.
The server can be software-based or a hardware solution that is capable of generating, storing, protecting, and distributing cryptographic keys.
Additional options for the server is services to support security. Examples of these services are audit services, backup/archive services, policy management, and access control services.[2]

**KMSS Ops**.
The server-to-server operations define a set of high level operations, messages and transport mechanisms that allow KM Servers to communicate with each other. These operations facilitate the communication and exchange of key management services between servers. With these operations a hierarchical key distribution schema can be build for example, or the availability of keying material can be raised.

**KM Admin**.
The Key Management administrator is responsible for configuring and maintaining the infrastructure. An example responsibility of an administrator is that when a new KM Client is added it needs to be registered in the KM Server before it can use generated keys from the server.
The arrows between KM Admin and KM Clients are examples, it does not mean that there are KM Clients that are not administered.

**KMCS Ops**.
The client-to-server operations define a set of high level operations, messages and transport mechanisms that allow KM Clients to communicate with KM Servers.
An implementation of the communication is a Remote Procedure Call (RPC) interface exposed by a KM Server. Request and response messages are formatted using the SOAP Document/Literal Wrapped pattern, and are transmitted over a TLS encrypted connection using HTTP 1.1.[2]
With these operations, requests and responses of for example keys and their attributes can be exchanged between clients and servers.

**KM Client**.
The KM Client consists of two components, an encryption entity and a storage medium.
The encryption entity is the component that can ask for and receive cryptographic keys, enforce policies and encrypt/decrypt data from a storage medium.
The storage medium is the component where the data is stored, it can be an optical (DVD), magnetic (tape) or solid-state (SSD) memory device.

# 6 Standards

There are different standards or standardisation efforts for the key management subject.
A list of the standards or efforts at the moment of writing are:

- ANSI X9 Financial Industry Standards
  `http://www.x9.org/home`
  Focused on the financial market.
- GlobalPlatform Key Management System
  `http://www.globalplatform.org/specifications.asp`
  Focused on smart cards and -devices.
- IETF Provisioning of Symmetric Keys (KEYPROV) Working Group
  `http://tools.ietf.org/wg/keyprov/`
  Focused on provisioning symmetric keys.
- ISO/IEC 11770: Key Management
  `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=19687`
  Focused on a general model for cryptographic key management.
- KeyGen2: Key Provisioning/Management Standards Proposal
  `http://keycenter.webpki.org/home`
  Focused on mobile phones, namely Google's Android.
- National Institute of Standards and Technology (NIST)
  `http://csrc.nist.gov/groups/ST/toolkit/key_management.html`
  Focused on recommendations for general cryptographic key management.
- OASIS Enterprise Key Management Infrastructure (EKMI) Technical Committee
  `http://www.oasis-open.org/committees/ekmi/`
  Focused primarily on symmetric keys.
- Trusted Computing Group: Infrastructure Work Group and Key Management Services Subgroup
  `https://www.trustedcomputinggroup.org/home`
  Focused on key management of TPM (Trusted Platform Module) devices.
- W3C XML Key Management (XKMS)
  `http://www.w3.org/2001/XKMS/`
  Focused on distributing and registering public keys working with XML Signatures and XML Encryption.
- IEEE P1619.3 Key Management
  `https://siswg.net/index.php?option=com_content&task=view&id=35&Itemid=76`
  Focused on key management for storage devices.
- OASIS KMIP (Key Management Interoperability Protocol)
  `http://xml.coverpages.org/ni2009-02-27-a.html`
  Focused on communication between devices in a key management infrastructure.

Not all of these standards are relevant to our research. We used the following criteria to narrow down our research scope:

- Standardisation efforts focusing on storage
- The standards must be open

By using these criteria we were left with two standards or efforts that we will be taking a closer look at:

1. IEEE P1619.3 Key Management[7]
2. OASIS KMIP (Key Management Interoperability Protocol)[8]

First off before we can take a closer look at the above standards a few pointers must be given for comparison.
To have some comparison pointers we thought of a couple of different building blocks that a Key Management System/Infrastructure must have, the building blocks are:

**Key Operations** These operations are needed to communicate key states between KM Clients and KM Servers and to send/receive keys.

**Policies** Policies are deliberate plans of action to guide decisions and achieve rational outcome. Key Management Policies are used to guide assignment, retention, key wrapping, replication and access control decisions on keys.[2]

**Namespace** This defines the unique identifiers that can be used to name and access specific keys.

**Auditing** When managing keys, logs must be kept to ensure the safeguard of the keys during and after their lifecycle.

In the following sections we will be discussing the different building blocks and their implementation in the different standards.

## 6.1 IEEE P1619.3

The standardisation effort of IEEE is trying to create a protocol that allows for secure interchange of keys between KM Clients with storage units and KM Servers. They are not yet working on a protocol to allow server-to-server communication.
By creating this standard they will avoid vendor-locking in the future.

### 6.1.1 Key Operations

We will start with the key operations.
Some of the basic operations that can be used between clients and servers are:

- Create/Generate Key
- Store Key
- Get Key

The Create/Generate Key operation makes a KEY object containing a list of attributes and policies. A key is generated in the KM Server, upon invocation of this operation, on the basis of templates or datasets.
Important attributes for the key lifecycle are the state expiration timers which are the T_EXPIRED and T_DISABLED, these timers are used to indicate the amount of time a key shall remain in a specific state. Another important attribute is the STATE attribute which indicates what is the current state of the key.

The Store Key operation gives the KM Client the possibility to store the keys generated at the client in the KM Server.

Get Key operation is invoked by a client to get a key from the KM Server. The client may invoke the key based on a unique identifier, name or attributes.
The server then returns the key based on the policies that govern the key use.[2]

### 6.1.2 Policies

There are different policies that can be used to govern key generation and use.
The policies that are implemented in IEEE P1619.3 are:

**Key Assignment Policy** This policy specifies the key scope when determining which type of data should be encrypted with a specific key. And when to generate new keys for key rotation.

**Retention Policy** With this policy key state timers are initiated and dictate the duration for which keys are accessible to a given client and when new data should no longer be encrypted with a given key.

**Wrapping Policy** This policy dictates if a key should be wrapped (encrypted) with a Key Encryption Key (KEK).

**Audit Policy** This policy states the auditing requirements that need to be enforced on keys and clients.

**Access/Distribution Policy** This determines which clients and key management servers may access which keys. The KM Server enforces this policy by checking in lookup tables storing keys to data assignments, and clients to data permissions.

**Caching Policy** The key caching policy dictates whether a key shall be cached by a KM Client and if so, the duration for which it is cached.

### 6.1.3 Namespaces

In IEEE P1619.3 a globally unique identifier is called a SO_GUID (Security Object Global Unique Identifier).
SO_GUID is a four level hierarchy, that contains the following information in hierarchical order:

1. SO_Family: A mandatory two-alphanumeric character code that describes the format for the following fields.
2. SO_Domain: An optional fully qualified domain name as defined in RFC 1034, that identifies a domain of keys under a SO_Family.[5]
3. SO_Context: An optional value that identifies a name space that is common across a set of keys in a SO_Domain.
4. SO_Handle: A mandatory value that is unique under the given SO_Context and corresponds to a specific KMS security object.

A valid SO_GUID can resolve to a Key Management Service (KMS) security object, a KMS symbolic link, or nothing.
Security objects are the fundamental objects maintained by KMS. The KMS security objects include keys, KM Client information, key groups, KM policies, and KMS logs.
The value of KMS symbolic link is a SO_GUID.
Since it is possible that a SO_GUID refers to an object that does not exist or that will exist in the future, a SO_GUID may resolve to nothing.[2]

An example of a SO_GUID that resolves to a key is, `km://domain/key/dir1/key123`.

- km: assigned URI namespace (SO_Family)
- domain: fully qualified DNS name (SO_Domain)
- key: this identifies an object namespace (SO_Context)
- dir1/key123: the path to the key (SO_Handle)

### 6.1.4 Auditing

The KM Server can keep logs and other information for auditing purposes.
Via the Audit Policy explained above, logs can be kept of key use and client access.
To ensure that the auditing information is kept on the KM Server, a Push Audit Message operation can be implemented in the KM Client.
This operation can sent log messages back to the server when cryptographic units access a key that is on the client.

## 6.2 OASIS KMIP

The standardisation effort of OASIS is primarily trying to create a standard communication protocol for between a KM Client that consumes keys and a KM Server that creates and manages keys.
By creating this standard they are trying to have a broader scope than the primarily storage-related IEEE P1619.3.

### 6.2.1 Key Operations

KMIP has over twenty-five operations that can be applied to keying material (KMIP calls these keys Managed Objects).
Of these twenty-five operations only the following seven have direct relevance to the key lifecycle, this is because they set the time that indicates a key state transition:

- Create
- Create Key Pair
- Activate
- Revoke
- Destroy
- Archive
- Recover

The Create operation generates a new key with some chosen and otherwise default attributes, sets the State attribute to Pre-Active and sets the Initial Date attribute.

The Create Key Pair operation generates a new public/private key pair with some chosen and otherwise default attributes, sets the State attribute to Pre-Active and sets the Initial Date attribute.

The Activate operation sets the State attribute to Active and the Activation Date attribute to the date and time when the key may begin to be used.

The Revoke operation sets the State attribute to Deactivated, optionally to Compromised when compromised and sets the Deactivation Date attribute or the Compromise Date attribute. When compromised the Compromise Occurrence Date attribute is set to the date the key was first believed to be compromised.

The Destroy operation indicates that a key may be destroyed, the server decides how and when this is done. When this happens the State attribute is set to Destroyed or Destroyed Compromised if the key was compromised and the Destroy Date attribute to the date and time when the key was destroyed.

The Archive operation indicates that a key may be put in archival storage. The key management system decides when and where this happens. When the key is put in storage the Archive Date attribute is set.

The Recover operation recovers an archived key from storage and deletes the Archive Date attribute.[9]

### 6.2.2 Policies

KMIP has four types of policies that can be set by a server:

**Operation Policy Name** An operational policy that indicates who may do which key management operations on the key. When this policy is not created with the key, a default policy is used depending on the type of cryptographic key.

**Cryptographic Usage Mask** A functional policy that indicates what cryptographic functions may be done with the key.

**Lease Time** A caching policy that indicates how long a client can keep a key, before it needs to renew its lease on the key.

**Usage Limits** Another operational policy that limits the use of key for protection purposes to a certain amount of objects or bytes.

### 6.2.3 Namespaces

KMIP doesn't identify the need for a namespace. All it does is create a Unique Identifier for a new key within a single key management system and denies any change or deletion of this Identifier. For import/export of keys between servers, KMIP recommends the use of a globally unique identifier.

### 6.2.4 Auditing

The KMIP documentation doesn't specify an audit procedure or policy, all it does is hint to the use of auditing in the following three situations:

1. When key's state become Destroyed Compromised: The object is no longer usable for any purpose, however it's compromised status may be retained for audit or security purposes.
2. The Revocation Message is an optional field in the Revocation Reason attribute of a Managed Object, which is used exclusively for audit trail/logging purposes and may contain additional information about why the object was revoked, for example "Laptop stolen" or "Machine decommissioned".
3. Result Message: This field may optionally be returned in a response. It contains a more descriptive error message, which may be used by the client to display to an end user or for logging/auditing purposes.[9]

## 6.3 Comparison

When looking at the differences between how IEEE P1619.3 and OASIS KMIP have setup their basic elements for a centralised key management system, it is clear that KMIP has put more effort in communication between client and server through requests/responses and less in defining strict policies, namespaces or audit procedures.

KMIP and P1619.3 both base their key management lifecycle on the lifecycle in NIST SP 800-57 explained in section 4 on page 8, but use it differently. Table 6.3 gives an overview of the differences in key states in the different standards compared to the NIST lifecycle.
An example of a difference is KMIP indicates the time a transition take place, P1619.3 gives the period after which a transition can take place. P1619.3 uses policies to determine the transition period and KMIP let's the client and/or server decide when to change states.

| **Management Phases** | **NIST SP 800-57**[6] | **IEEE P1619.3**[2] | **OASIS KMIP**[9] |
|---|---|---|---|
| *Pre-Operational* | Pre-Activation | Pre-Activation | Pre-Active |
| *Operational* | Active | Protect-and-Process<br>Process-only | Active |
| *Post-Operational* | Deactivated | Expired<br>Disabled | Deactivated |
| | Compromised | Compromised<br>Disabled-Compromised | Compromised |
| *Destroyed* | Destroyed | Destroyed | Destroyed |
| | Destroyed Compromised | Destroyed-Compromised | Destroyed Compromised |
| | | Purged | |

Table 1: Comparison between Key Management Lifecycles

P1619.3 separates responsibilities from client and server, where KMIP relies on the users.
This is also the case when looking at namespaces, KMIP doesn't set restriction for unique identifiers and just recommends using globally unique identifiers. On the other hand the namespace of P1619.3 uses a strict format.
Auditing is an element that both have thought about and agree on how it needs to be used, but haven't fully developed it.
P1619.3 is looking at the possibility to work together with KMIP so that the standards can mutually benefit from the their own views on cryptographic key management.

This comparison shows that P1619.3 seems to be better in defining a standard than KMIP, but KMIP stated to only want to standardise communication between KM Servers that create and manage keys and KM Clients that request and use these keys.
KMIP's main focus is interoperability between KM Servers and KM Clients. This means KMIP has a broader scope than only encrypting data in storage devices as is the focus of P1619.3.

# 7 Implementations

Like the standards discussed above, there are different implementations of key management systems.
These implementations are not based on any standard, that is why we chose to only discuss further the one that we think will have more influence on the standardisation efforts at the moment. The implementations at the moment are:

- EMC's RSA Key Manager
  `http://www.rsa.com/node.aspx?id=3013`
  This is a software-based solution to generate and manage symmetric keys used in applications.
- NetApp Lifetime Key Management
  `http://www.netapp.com/us/products/storage-security-systems/lifetime-key/`
  A hardware appliance that can backup and distribute keys to encrypt/decrypt data.
- Sun Key Management System
  `http://opensolaris.org/os/project/kmsagenttoolkit/`
  Open-source implementation of a KM Client that interfaces with a Key Management Appliance to encrypt storage.
- StrongKey
  `http://www.strongkey.org/`
  Open-source symmetric key management system for application encryption

By narrowing our research, to implementations that are focused on storage and are open-source, we chose Sun Key Management System (Sun KMS) to take a closer look at.

## 7.1 Sun Key Management System

Sun has been using it's key management protocol in it's tape drives, recently they made their protocol open-source by releasing the key management API (Application Programming Interface). Coincidentally Sun released it's API a week after the KMIP efforts were released by a competitive group in the key management industry.[10]

Sun implemented key management in the following manner, Sun KMS uses so called KMS clusters as KM Servers. The cluster consists of one or more interconnected Key Management Appliances (KMA).
These appliances manage the keys and also store them, this information is replicated over the cluster.
The KM Client is an API that can interact with the KMS cluster for creating and obtaining keying material.

By using the same pointers described in section 6 on page 12, we can analyse different components of this system.

### 7.1.1 Key operations

Two of the basic key operations are:

- CreateKey
- RetrieveKey

The CreateKey operation creates a new key and sets it's key state to activated and associates it with a specific KeyGroup. In this KeyGroup the keys are organised and given KeyPolicies. The policies provide settings for the periods that a key can remain in a given key state.

RetrieveKey is an operation that retrieves a key according to it's Key ID, optionally a KeyGroup can be specified.

### 7.1.2 Policies

Sun KMS knows two policies, an access policy called KeyGroups and a retention policy called KeyPolicy.
The KeyGroups organise keys and associate them with a KeyPolicy. KeyGroups are also used to enforce access to the keying material by the KM Clients.
The KeyPolicy provides the timers that are applied to the keys that dictate the key state changes.

### 7.1.3 Namespaces

There is no specified unique namespace for the keying material in the Sun KMS.
Because the information in the KMS cluster is identical, it doesn't enforce uniqueness on all KMA.[11]

### 7.1.4 Auditing

The KM cluster maintains a log of all auditable events occurring throughout the system.
KM Clients may contribute entries to this log by generating a KMS audit log entry.

## 7.2 Sun KMS, KMIP and P1619.3

When you have a look at the differences between the KMIP and P1619.3 standards and the Sun KMS implementation you can see that the standards include more different parts than the implementation.
Where KMIP and P1619.3 try to explain every detail of how they believe key management should be implemented, Sun KMS gives a more global overview and tries to explain what they think is most important for key management.

When comparing the basic elements we see that Sun uses the same NIST SP 800-57 guidelines as the two standards.
Just as with P1619.3 it uses a duration for the key states instead of a state transition time, also it follows the separation of responsibilities by proper use of policies for key access and retention.

Sun KMS doesn't specify a namespace, which could make interoperability difficult if other vendors implement their namespace a different way. Sun KMS addresses this problem by having every KMA contain the same information. This provides good availability and failover when a specific KMA isn't available.
Sun is planning on proposing his key management implementation as a reference implementation for the KM Client of P1619.3.

Finally the audit capabilities show that Sun KMS has been able to implement it well, where KMIP and P1619.3 are still trying to decide on how to get this right.

# 8 Conclusions

Cryptographic Key Management is a hot topic at the moment, there are different organisations busy with standardisation efforts or implementations. Because there is no head organisation at the moment that stipulates how key management should be, it is difficult to accurately pinpoint what the future will look like for key management.

Like it was hinted in section 7.2 Sun is looking for a chance to push it's key management implementation in the standardisation efforts at the moment. Hopefully it will not negatively influence the standards that would come forth from that relationship. In the past there were standardisation efforts that were negatively influenced by the industry, we hope that this will not be one of them. On the other hand P1619.3 is looking at the possibility for coöperation with KMIP. This can have the effect that all these three organisations will work together. If this happens it can positively influence the course of key management standardisation.

When one standard is widely accepted by the industry, the interoperability problem between different vendors can be solved. This can increase the scalability possibilities and raise the availability between different organisations, this also opens up the possibilities of creating hierarchical key management infrastructures.

We find that key management is very confusing at the moment, we hope this will improve in the future. This can be achieved by one clear standard that incorporates all components of a key management system.

# References

[1] Bob Lockhart. Key management services provisioning systems for enterprise key management. Presentation, September 2008. http://keymanagementsummit.com/2008/presentations/Lockhart,%20Bob%20-%20Provisioning%20System%20for%20Enterprise%20Key%20Management%20and%20Distribution.pdf.

[2] Security in Storage Working Group. Draft standard for key management infrastructure for cryptographic protection of stored data. Technical report, IEEE, February 2009. P1619.3/D6.

[3] Tim Hahn. Cryptography everywhere a security administrator's nightmare. Presentation, September 2008. http://keymanagementsummit.com/2008/presentations/Hahn,%20Tim%20-%20KMS-2008-Presentation-20080827.pdf.

[4] Walt Hubis. Key management: Getting a lock on secure storage. Website, February 2008. http://www.enterprisenetworksandservers.com/monthly/art.php?3520.

[5] Landon Curt Noll. Key management services an overview. Presentation, September 2008. http://keymanagementsummit.com/2008/presentations/Noll,%20Landon%20-%20Key%20Management%20Services%20-%20v109.pdf.

[6] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. Recommendation for key management part 1: General (revised). In *NIST Special Publication 800-57*. March 2007. http://csrc.nist.gov/publications/PubsSPs.html.

[7] IEEE. P1619.3: Key management. Website. https://siswg.net/index.php?option=com_content&task=view&id=35&Itemid=76.

[8] OASIS. Oasis kmip tc. Website. http://xml.coverpages.org/KMIP-TC-Proposal.html#charterProposal.

[9] Robert Haas. Kmip - key management interoperability protocol. Technical report, OASIS, February 2009. Draft version 0.98.

[10] Beth Pariseau. Sun jumbles key management picture. Website. http://searchstorage.techtarget.com/news/article/0,289142,sid5_gci1348276,00.html.

[11] Sun Microsystems Inc. Sun crypto kms agent toolkit. Program documentation.