Implementation and Operations Guidelines for an Enterprise Key Management Infrastructure (EKMI)

Table of Contents

1. Introduction

For the last few decades, enterprises have been protecting their sensitive data assets from attacks with a variety of technical and procedural counter-measures. Many technologies have been invented and deployed to address the security problem, most notably the Firewall that blocks unauthorized access to network ports and services on computers and/or networks, intrusion detection systems, virtual private networks, two-factor authentication, etc. Companies have invested billions in implementing such technologies.

However, as press reports indicate, companies have been plagued with devastating attacks against their information technology assets, disrupting systems and breaching sensitive information of millions of US-based residents. Despite rising levels of investment in security-related technology, the losses and attacks continue to mount.

For too long, IT organizations have been focused on protecting the network and computer perimeter, rather than the actual data itself. This is because:

• It was significantly easier to protect one gateway to the corporate network than hundreds of applications or thousands of computers;
• The types of attacks were not sophisticated enough to warrant any more protection than a firewall and basic access control measures – unless you happened to be in defense or finance-related organizations;
• Business demands on IT were growing very rapidly requiring attention from IT organizations to deliver information processing capability than to focus on data-protection;
• While sophisticated data-protection technologies – encryption, cryptographic hardware modules, etc. - existed, they were difficult to use & manage, and expensive;

Enterprises are coming to the realization that despite the network and computer-based controls, the actual data itself has been left unsecured, and attackers are finding ways to get to it. As a result they need to start encrypting sensitive data wherever it is stored. Additionally, many new regulations – while not mandating it (with the exception of the Payment Card Industry’s Data Security Standard) – are increasingly legislating that sensitive data be encrypted to avoid lawsuits.

However, the distributed nature of client-server computing has made this task very complex. Sensitive data is no longer contained within a single mainframe-like operating environment that is under the control of professionally trained IT staff. It is spread out across dozens – if not hundreds or thousands of computers – across the globe. Laptops, flash drives, offshore development facilities, outsourced Disaster Recovery data centers, offline storage farms, etc. These are just some of the many places that sensitive data might exist.

Encrypting sensitive data consistently, across diverse devices, operating environments and applications is a monumental task. It has been almost impossible to accomplish this, because there did not exist any products, protocols or guidelines to perform this activity with such a far-reaching scope.
Until now.

The **OASIS Enterprise Key Management Infrastructure Technical Committee (EKMI TC)** has taken up the gauntlet, and with the support of many existing technological standards and the royalty-free donation of a recently-created open-source software product's protocol, for the first time enterprises will have a consistent way of encrypting sensitive data consistently across all their major operating environments and devices, in a secure, open and cost-effective manner.

The rest of this document describes an **Enterprise Key Management Infrastructure (EKMI)**, its components and provides guidelines for establishing a standard EKMI. However, enterprises deploying EKMIs must take their own business, security and operational requirements into consideration before implementing this recommended architecture.

### 1.1. Background

#### 1.1.1. What is key-management?

Key-management is the term given to the discipline of managing cryptographic encryption keys. As with all computer-related objects, cryptographic keys have a life-cycle that can be simple for ad-hoc use, but complex in large and sophisticated environments. This life-cycle consists of one or more of the following:

- **Key Generation** – the act of generating cryptographic keys for use within applications;
- **Key Escrow** – the act of securely storing a cryptographic key so it can be retrieved for later use;
- **Key Recovery** – the act of retrieving a cryptographic key securely, typically for decrypting ciphertext (encrypted data);
- **Access Control** – the activities involved in ensuring that cryptographic keys are accessible only to authorized entities;
- **Key Destruction** – the act of securely destroying a cryptographic key at the end of its useful lifetime, so it is inaccessible to anyone thereafter.

#### 1.1.2. Why key-management?

Every application that provides cryptographic services – securely - must perform at least the first four key-management functions – generate, escrow, recover and manage access. Consequently, any application that performs these functions, is providing key-management.

What distinguishes ad hoc key-management performed by business applications and sophisticated key-management applications is scale. While simpler tools such as the Java Keytool, OpenSSL, the Netscape Security Service (NSS) Library and GnuPG perform admirably for a few dozen to, perhaps even hundreds of keys, in large-scale environments where, potentially hundreds of thousands – perhaps even millions – of cryptographic keys must be managed, these tools are inadequate; professional and sophisticated key-management software is essential to make these tasks easier to manage.

As companies discover their need to encrypt sensitive data across diverse devices and operating environments, they are starting to discover the complexities of managing encryption keys across the enterprise. In the words of Trent Henry, a security analyst at the Burton Group:
“The life cycle of encryption keys is incredibly important. As enterprises deploy ever-increasing numbers of encryption solutions, they often find themselves managing silos with inconsistent policies, availability, and strength of protection. Enterprises need to maintain keys in a consistent way across various applications and business units,” said Trent Henry, senior analyst, Burton Group. “EKMI will be an important step in addressing this problem in an open, cross-vendor manner.”

1.2. What is an EKMI?

There are two types of cryptographic keys – symmetric and asymmetric. Symmetric cryptographic keys are used with algorithms where the same key is required for encryption and decryption of data. Asymmetric cryptographic keys are different in that one key – typically, the Public Key – is used for encryption of data, while another related key – the Private Key – is used for decryption. In the use of Digital Signatures, it is the exactly opposite, where the Private Key is used to encrypt data while the Public Key is used to decrypt it.

Historically, symmetric and asymmetric keys have been managed separately because of their distinctive characteristics. In a large-scale environment, asymmetric keys have been managed within Public Key Infrastructures (PKI) while symmetric keys have been managed within the applications that performed symmetric encryption.

This creates a problem. While an enterprise can have a single infrastructure – a PKI – for managing asymmetric keys, it is currently forced to manage many key-management infrastructures with the use of symmetric keys. Not only does this lead to a higher total cost of ownership (TCO) due to distinct operational environments, distinct procedures, training, audits, etc., but due to the large number of key-management infrastructures, it is possible to introduce a vulnerability within the system due to human errors or errors of omission.

It is the OASIS EKMI TC’s belief that all cryptographic keys need to be managed within a single infrastructure – an Enterprise Key Management Infrastructure (EKMI).

An EKMI is defined as a collection of policies, procedures and technology to manage all cryptographic keys in the enterprise.

It primarily consists of two major components: a PKI and a Symmetric Key Management System (SKMS) for managing asymmetric and symmetric cryptographic keys, respectively. The rest of this document describes these two major components of an EKMI and provides guidelines in implementing an optimal EKMI architecture.

Current computer technology requires that symmetric and asymmetric keys must be managed distinctly. However, it is the goal of the OASIS EKMI TC to work towards consolidating these technologies so that a single infrastructure is capable of providing both types of key-management services to any authorized entity on the network.

It is currently focused on standardizing the SKMS component of EKMI – since the PKI component is already well standardized by the International Standards Organization (ISO) X.509 and the Internet Engineering Task Force (IETF) Public Key Infrastructure X.409 (PKIX) Working Group.

Once the SKMS protocols and guidelines have been standardized, the TC expects to begin work on the consolidation of PKI and SKMS.
1.3. Components of an EKMI

1.3.1. PKI

1.3.1.1. What is a PKI?
1.3.1.2. Why PKI?

1.3.2. SKMS

1.3.2.1. What is an SKMS?

A Symmetric Key Management System (SKMS) is a collection of policies, procedures and technology to manage symmetric cryptographic keys used with encryption algorithms.

The primary characteristics of an SKMS, as defined by OASIS, are that it:

- Provides a single infrastructure to define policies for symmetric key management;
- Provides a single infrastructure to manage all symmetric cryptographic keys used for encryption;
- Consists of at least one or more Symmetric Key Services (SKS) servers that provide symmetric key-management services;
- Uses a Symmetric Key Client Library (SKCL) on SKMS clients to interact with the SKS servers;
- Uses the Symmetric Key Services Markup Language (SKSML) as the protocol between SKMS clients and SKS servers for communications;
- Is platform and application independent, so that any authorized client that implements the SKSML protocol may receive key-management services;
- Is scalable to service millions of SKMS clients on the network;
- Permits the use of symmetric keys on SKMS clients even when all SKS servers are unavailable on the network;
- Is extremely secure;

1.3.2.2. Why SKMS?

As companies start to encrypt data across the enterprise to protect sensitive data from attacks and to comply with regulations requiring encryption, it is imperative that IT organizations have full control of symmetric encryption keys used by applications, regardless of where it is used within the enterprise.
An SKMS that conforms to the above-mentioned characteristics provides IT organizations with that control. It allows them to manage the policies, keys and access to keys in one place within this infrastructure.

With multiple key-management systems IT organizations will have an environment that resembles the following diagram:

However, with an SKMS, IT organizations will be able to streamline their use and management of symmetric encryption keys, resulting in an efficient environment that resembles the following:

An environment with an SKMS leverages computing power across the enterprise to encrypt data, while providing the enterprise with the appropriate control to manage the encryption keys centrally.

2. PKI
   2.1. Applications
   2.2. Architecture
2.3. Policy Considerations
2.4. Best Practice Policies for PKI
2.5. Security Considerations
2.6. Technology
   2.6.1. Certification Authority
   2.6.2. Registration Authority
   2.6.3. Hardware Security Module(s)
   2.6.4. Publishing Directory
   2.6.5. Certificate Validation
2.7. Standards and Protocols
2.8. Operations

3. SKMS

3.1. Applications

An SKMS can be used in many applications. Some common examples where an SKMS can be used in different industries are:

Retail

A retail company with hundreds of stores and thousands of Point-of-Sale (POS) terminals, accepts credit cards, whose Credit Card Numbers (CCN) must be encrypted to comply with the Payment Card Industry Data Security Standard (PCI-DSS). The encrypted CCN and the decryption key must be available to Store employees and Customer Support Representatives (CSR), the back-office processing database and application environment, the bank-settlement application and the fraud analysis application.
With a **Symmetric Key Management System (SKMS)** and a **Symmetric Key Services (SKS)** server on the retailer's network, the POS terminal application can securely request and receive a symmetric encryption cryptographic key from the SKS server using SKSML, encrypt the CCN and store the encrypted “ciphertext” with the symmetric key's **Global Key ID (GKID)** in the application's database. The symmetric key itself may now be discarded by the application or securely cached for future use on the POS terminal, based on the company's policy.

Any authorized application on the retailer's network – the back-office database, the bank-settlement application, any other POS terminal, the CSR application – may request the same symmetric key from the SKS server using the GKID, and upon receiving it decrypt the ciphertext to recover the “plaintext” CCN.

Because the symmetric key and its meta-data are securely generated and stored on the SKS server even before a client uses the key, the company is assured of recovering the decryption key by any authorized entity using the SKSML protocol at any time.

While enterprises can perform all these functions today, it must be re-emphasized that an SKMS allows managing the encryption keys across all these applications in one place.

**Healthcare**

**Emergency Medical Technicians (EMT)** respond to medical emergencies and collect sensitive medical data on laptop computers. The **Health Insurance Portability and Accountability Act (HIPAA)** requires this information to be maintained confidentially, while doctors and nurses in the **Emergency Room (ER)** must be able to access this it without any hindrance.

With a **Symmetric Key Management System (SKMS)** and a **Symmetric Key Services (SKS)** server on the healthcare provider's network, the laptop application can securely request, receive and cache multiple symmetric encryption cryptographic keys from the SKS server using SKSML, even before using the keys to encrypt any medical data. This ensures that the EMT can encrypt/decrypt data even if they do not have access to the network.
Additionally, all EMTs can be consolidated into a “KeyGroup” within the SKS server, and ER doctors and nurses can be authorized to retrieve any symmetric key created at the request of any member of the EMT KeyGroup. This ensures that when the EMT encrypts medical data and transmits it to the hospital, perhaps over wireless network even before getting the patient there, doctors and nurses can receive the ciphertext and decrypt the information as soon as it arrives. They would be able to access the same symmetric key used by the EMT in the field, and escrowed in the SKS server, through ACLs established in advance – even for keys that have not yet been generated.

**Government**

The US Department of Homeland Security (DHS) operates a program known as the Protected Critical Infrastructure Information (PCII) Program. This program is designed to encourage private industry to share its sensitive security-related business information with the US Federal government.

However, a problem arises in that the private industry would like to see its data maintained confidentially on DHS systems, while the DHS needs to share it with appropriate First Responders and other security personnel at the Federal, State and City level.

With a Symmetric Key Management System (SKMS) and a Symmetric Key Services (SKS) server on the DHS network, private companies working with the DHS on this program could request and receive symmetric encryption keys using SKSML and transport the encrypted ciphertext to DHS in any means suitable to them. The symmetric key itself may now be discarded by the private company as the DHS may establish a policy of using such keys for a single encryption only.

With the ciphertext in the DHS databases, and the symmetric key in the DHS SKMS, any authorized entity working with the DHS that has access to the ciphertext, may be granted access to retrieve the symmetric key. The authorized entity – be it a DHS Agent, or personnel at the Federal, State, County or City level - can use SKSML to request the same symmetric key to decrypt the necessary information to perform their tasks.
Finance

Banks, Credit Unions and other financial institutions with Automated Teller Machines (ATM) have a need for protecting the Personal Identification Numbers (PIN) of their customers during ATM transactions and in communications with the financial institution's network.

Traditional symmetric key management requires physical distribution mechanisms to deliver key packages (either in the clear or encrypted with a key-encryption key that is identical across their network) to the devices in the network. In a highly distributed global network, this creates a resource-intensive environment to create, package, deliver and supersede the key material in use. In many circumstances, symmetric keys may be used for a period longer than recommended, due to the expense of physically replacing the symmetric key-material.

With a Symmetric Key Management System (SKMS) and a Symmetric Key Services (SKS) server on the financial institution's network, all ATMs can use SKSML to request and receive symmetric encryption keys securely over the network without having to send personnel to the ATM. Additionally, because of the message-level security inherent in SKSML, they do not need a private or secured network to communicate these keys – they can be transported over the internet while maintaining the same level of security in the message as Secure Socket Layer (SSL), Transport Layer Security (TLS) or Internet Protocol Security (IPSec).

Secondly, the symmetric keys delivered to the ATMs over the network are encrypted with unique transport keys, thus ensuring that a compromise on any one ATM would not affect any other ATM on the network.

Financial institutions can even establish policies on the SKS server to rotate symmetric encryption keys daily, or even hourly on the ATM, if desired since the process of delivering the symmetric encryption key to the ATM is now completely automated.

Another application in the financial industry is the sharing of sensitive financial information by banks/investment houses with retail financial consultants.
Currently, many financial institution's that produce sensitive financial data about customers from corporate databases/applications, share this data with their partners using the File Transfer Protocol (FTP)-based system. Security is, usually, layered on this either through the use of standalone encryption tools such as GnuPG or the use of Secure FTP (SFTP). What is cumbersome about this architecture is that the encryption keys in GPG or SFTP must be managed manually, even if the files are automatically encrypted by the corporate applications for the specific recipient.

With a **Symmetric Key Management System (SKMS)** and a **Symmetric Key Services (SKS)** server, the financial institution's Corporate Database/Application can use SKSML to request and receive symmetric encryption keys to encrypt the financial data, and then place them on a web-server that is accessible over the internet.

The web server does not need any special protection – such as authentication or SSL/TLS because all datafiles are encrypted on the web server. Additionally, the decryption keys are never stored on the web server or embedded in the encrypted files. Only references to the Global Key ID are stored in the encrypted file.

Once the Corporate Database/Application has placed the encrypted file on the web server, the recipient is sent the URL of the file in an e-mail. Once again, because of the encrypted contents and the separation of the decryption key from the ciphertext, the e-mail does not require any special protection.

The Partner can now download the file using a standard browser, or an HTTP-enabled application that is capable of downloading the file and requesting the symmetric key from the SKS Server to decrypt the file. If using the browser, the Partner will also be provided with a decryption utility that automatically requests the decryption key from the SKS server. Preconfigured Access Control rules on the SKS server will permit only specific authorized Partners to access specific decryption keys, so one Partner may never decrypt a file intended for another.

In this manner, financial institutions with an SKMS, can replace their file-transfer and manual key-management infrastructure with standard web servers, browsers and automated key-management solution.

**E-Commerce**

E-commerce portals that accept credit cards for transactions must not only comply with PCI-DSS, but must ensure that their 24-hour internet-facing portals do not reveal
sensitive CCN to attackers that may have taken control of their web, portal or application server.

With a **Symmetric Key Management System (SKMS)** and a **Symmetric Key Services (SKS)** server on the e-commerce vendor's network, all servers and databases within their e-commerce applications can use SKSML to request and receive symmetric encryption keys to protect CCN and/or other sensitive data from unauthorized access.

Because all applications within the e-commerce infrastructure are part of the same SKMS, they do not need to transport the decryption key to other servers on the network, even as they transport the ciphertext to them. The server down the fulfillment-chain will have access to the same symmetric key from the SKS server using SKSML.

**Corporate**

Large corporations need to protect sensitive data on tens of thousands of laptops, Personal Digital Assistants (PDA), databases, and servers across their global infrastructure. **Chief Security Officers (CSO)** at these companies would like to define policies centrally, on how symmetric encryption keys can be used and managed across the enterprise.
With a **Symmetric Key Management System (SKMS)** and a **Symmetric Key Services (SKS)** server on the corporation's network, all laptops, PDAs, databases, desktops and servers can use SKSML to request and receive symmetric encryption keys to protect sensitive data from unauthorized access.

Even if laptops and PDAs are stolen, and databases breached, the CSO will not have to worry about disclosing such breaches to the press because the SKSML protocol allows for secure caching of symmetric keys on the PDAs and laptops. The mobile devices can have the same level of security in protecting data and symmetric keys as desktops and servers on the corporate network.

**Education**

Schools and Universities have multiple challenging requirements. On the one hand, they receive and must protect personal identification information (PII) of students, their parents and guardians from financial aid applications, and if they have a medical school and hospital associated with the university, the medical information of their patients. While on the other hand, they need to create an open network environment that fosters creativity and collaboration to enable their students to learn from their teachers, peers and resources on the internet.
With a **Symmetric Key Management System (SKMS)** and a **Symmetric Key Services (SKS)** server on the university's network, they can address these diverse interests by having the applications dealing with sensitive data use SKSML to request and receive symmetric encryption keys to protect this information. The Student Database in the Registrar/Bursar's office, the Faculty Database in Human Resources, the Patient Database in the medical school (if any), etc. can all share the services of an SKS server on the network to protect sensitive information.

Even if the student network was directly connected to the university's “business” network, sensitive data would still be secure since the students would not have the appropriate credentials to make requests to the SKS server. The SKSML protocol carries digital signatures in its header; therefore the SKS server has strong assurance of an authentic request from a client, and every client has strong assurance that it is not being attacked by someone masquerading as an SKS server.

### 3.2. Architecture

While there are many different ways in which an SKMS can be implemented, this document will describe a **Basic SKMS Architecture (BSA)** to serve a mid-to-large sized enterprise. The architecture will accommodate for redundancy and for security. Implementers of SKMS are expected to balance this Basic SKMS Architecture against their business, security and operational requirements and make adjustments accordingly.
This document will use the following diagram to describe elements of the BSA:

![Diagram of BSA network architecture](image)

The BSA assumes that the enterprise has a common network architecture with a Primary Data Center (PDC), a Disaster Recovery Data Center (DRDC) and a Demilitarized Zone (DMZ) all connected together on the company's Intranet. The Intranet is connected to the Internet through the DMZ, protected by the usual firewalls, etc.

Since the EKMI consists of both a PKI and the SKMS, both components are shown in the BSA; however, this section focuses only on the SKMS – the Basic PKI Architecture (BPA) is described in the PKI section of this document.

### 3.2.1. Primary Data Center (PDC)

The PDC is expected to have at least one SKS server to provide key-management services. If this is the first SKS server in the enterprise, this is designated the Global Symmetric Key Services (GSKS) server.

There are two primary differences between GSKS servers and normal SKS servers:

1. All SKMS policies, clients, servers, groups and access control rules – also known as SKMS meta-objects - are defined at the GSKS server; i.e. the SKMS Administrator logs into the GSKS to perform all administration tasks. The SKMS meta-objects are then pushed out to all SKS servers for enforcement.

2. All keys generated and escrowed at any SKS server in the enterprise are replicated to the GSKS server so that the enterprise has at least one repository where all symmetric keys are available.

Other than these two differences, GSKS servers function like normal SKS servers and provide the same key-management services.

All SKMS clients must point to at least one GSKS in their configuration files. This is no different from configuring Domain Name Service (DNS) clients where the client must be pointed to at least one DNS server for IP name-resolution. In
practice, in a configuration that consists of many SKS servers, SKMS clients will be pointed to the nearest SKS server in their client configuration, with the GSKS server as the server of last resort. In the BSA, the GSKS server will be one of two SKS servers in the client configuration files; the second server will be a GSKS-DR (see the section on Disaster Recovery Data Center for details).

While the BSA specifies at least one SKS server - a GSKS - for the PDC, real-world implementations may have as many SKS servers as the enterprise needs to accommodate its requirements. Companies may implement many SKS servers for performance reasons, redundancy reasons, or both.

The PDC is also expected to have the primary PKI environment that manages digital certificates for the SKMS so that SKMS clients can avail.

### 3.2.2 Disaster Recovery Data Center (DRDC)

The DRDC is expected to have at least one GSKS Server running in Disaster Recovery (DR) Mode (GSKS-DR) to serve up limited symmetric key services in the event the PDC becomes unavailable during a disaster.

The difference between a normal GSKS server and a GSKS server running in DR Mode (GSKS-DR) is that the latter cannot be used to create and/or update SKMS meta-objects and generate new symmetric keys. It can only be used to serve up existing SKMS meta-objects to SKMS servers, and existing symmetric keys to SKMS clients on the network.

The GSKS in the PDC is configured to replicate all its objects to the GSKS-DR so that the GSKS-DR has all objects known to the GSKS in the PDC.

### 3.2.3 De-Militarized Zone (DMZ)

The BSA recommends a standard web-proxy (HTTP proxy) for providing SKMS services to external partners and customers.

Many enterprises share data with partners and customers, some of which may require encryption. Current schemes for secure data-sharing with external entities typically consist of: i) an application that generates the shared plaintext (unencrypted) data; ii) an infrastructure to manage the encryption keys; and iii) an infrastructure to manage the file-transfers.

With the BSA it is possible to have the source application encrypt data directly into ciphertext with a symmetric key supplied by an SKS server, and place the encrypted file on a web-server where it can be downloaded using the HTTP protocol (see the Financial sector Use-Case earlier in this document)

The recipient, once having retrieved the file will need the symmetric key to decrypt the file. However, because the SKS server is within the Intranet of the enterprise, the external entity will not be able to reach the SKS server directly. Because the SKSML protocol - a Simple Object Access Protocol (SOAP)-based XML protocol - uses message-level security using XML Signature and XML Encryption, it only needs standard HTTP to transport requests and responses securely.

As a result, the web-proxy is all that's needed to have an external entity make a request for the symmetric key. The proxy having received the request, forwards the request to the SKS server inside the Intranet, and the response from the SKS server back to the original requester.

While it is entirely up to the discretion of the implementing company, it is reiterated here that the web-proxy does not need any transport-level security – such as Secure Socket Layer (SSL), Transport Layer Security (TLS) or IPSec – to...
protect the messages between the client and servers within an SKMS.

3.2.4. **(Global) Symmetric Key Services (SKS) Server**

The SKS server – or the GSKS server – is typically a Linux, UNIX or Windows-based server running a Java 2 Enterprise Edition (J2EE) Application Server and a Relational Database Management System (RDBMS). The Application Server and the RDBMS are expected to operate in the same address-space of the server, so no external inter-process communication (IPC) mechanism or security is necessary. However, if the Application Server and the RDBMS are separated physically and operate on distinct machines, their external IPC must be established and secured separately (see the section on Security for more details).

The BSA recommends using a cryptographic Hardware Security Module (HSM) on all SKS servers to protect the Private Keys of the Signing and Encryption certificates of the server, the symmetric keys being generated and used, and the cryptographic operations from being snooped on the system. The HSM may be in the form of a Trusted Platform Module (TPM) chip on the motherboard of the server, a smartcard, a PCI-based card or a network-attached HSM; the choice of implementation depends on factors such as speed, security, convenience of operations, etc. and must be assessed by implementers before the choice is made. However, except for Development and Test environments, it is not recommended that an SKS server operate without an HSM (see the Security section for more details).

The GSKS server is where all SKMS meta-objects are defined: servers, clients, groups, policies, access control rules, etc. Only one SKS server must be configured on the network as a GSKS server, because this server will replicate all its meta-objects to other GSKS-DR servers on the network. The BSA only requires one GSKS and one GSKS-DR in its implementation; however enterprises are encouraged to review their operational and availability requirements and add as many GSKS-DR servers to their implementation as they deem necessary.

3.2.5. **SKMS Client**

Any application that needs to encrypt data and which, consequently, needs key-management services, is an SKMS client. These can be:

- Databases storing sensitive information;
- E-commerce servers and Point-of-Sale (POS) registers accepting Credit Card Numbers (CCN) from customers;
- Automated Teller Machines (ATM) accepting bank cards from customers;
- Desktop PC's at Nurse's Stations dealing with patient data;
- Laptops
- Payroll applications;

An SKMS client can be written in most high-level programming languages such as C, C++, Java, RPG, etc. SKMS clients link in the **Symmetric Key Client Library (SKCL)** implemented for their platform and programming language. The SKCL, in turn, takes care of all communication between the SKMS client and the SKS server(s) for any key-management services. The SKCL manages a **key-cache** on the SKMS client, so that it can continue to encrypt and decrypt sensitive data even if all SKS servers are unavailable to it on...
the network. Policies that define how the SKCL uses the key-cache are defined centrally on the GSKS. The SKCL automatically requests key-cache policy information upon startup and keeps it updated at periodic intervals.

The BSA recommends using a cryptographic Hardware Security Module (HSM) on SKMS clients to protect the Private Keys of the Signing and Encryption certificates of the client. The HSM may be in the form of a Trusted Platform Module (TPM) chip on the motherboard of the client, a smartcard, a PCI-based card or a network-attached HSM; the choice of implementation depends on factors such as speed, security, convenience of operations, etc. and must be assessed by implementers before the choice is made. However, except for Development and Test environments, it is not recommended that an SKMS client operate without an HSM (see the Security section for more details).

3.3. Policy Considerations

There are two types of explicit policies within an SKMS: policies related to what types of symmetric keys are created and how they are used (KeyUsePolicies); and policies related to caching of symmetric keys on SKMS clients (KeyCachePolicies).

An SKMS conforming to the OASIS reference implementation will support KeyUsePolicies that allow for defining:

- What type of symmetric keys are generated (3DES, AES, etc.);
- What size keys are generated (128, 192 or 256-bit keys);
- How long a symmetric key may be used;
- For how many encryption transactions can a symmetric key be used;

An SKMS implementation may also support many optional attributes related to KeyUsePolicies such as:

- What types of applications are allowed to use a specific symmetric key;
- Where (geographical location) can a symmetric key be used;
- What time of day can a symmetric key be used;
- Labeling a key in accordance with a classification taxonomy;

An SKMS conforming to the OASIS reference implementation will support KeyCachePolicies that allow for defining:

- Whether an SKMS client should first look in its key-cache or contact the SKS server for a symmetric key;
- How many new (unused for any encryption transaction) symmetric keys is an SKMS client allowed to cache locally;
- How long can new symmetric keys be cached on the SKMS client;
- How many used symmetric keys is an SKMS client allowed to cache locally;
- How long can used symmetric keys be cached on the SKMS client;

There are no general rules about what policies are optimal for an implementation. Enterprises must evaluate their business, operational and security requirements in determining the appropriate KeyUse and KeyCache policies for their environment. In some cases, over-riding company policies may dictate how such technical policies are implemented in the SKMS.
3.3.1. Best Practice Policies for SKMS

There are many SKMS policies that implementers must take into consideration. They are:

- **Key-type and key-strength**
  - In this policy, enterprises must determine which type of symmetric encryption algorithm they wish to use for specific applications/uses, and what size key do they wish to use. The OASIS SKMS BSA supports the use of both **Triple Data Encryption Standard (3DES)** and the **Advanced Encryption Standard (AES)** and allows an enterprise to mix and match algorithms too, if desired.

  Note: While there are other popular encryption algorithms in use in the marketplace, the TC has specifically chosen to support the standards currently supported by the **World Wide Web Consortium’s (W3C) XML Encryption** standard. As this W3C standard evolves, so will the support of those algorithms by the EKMI TC.

  The reference implementation supports the 192-bit 3-DES standard, and the 128-bit, 192-bit and the 256-bit implementations of AES. While a conservative key-management policy might dictate AES-256 for all symmetric encryption uses, enterprises must evaluate their business, operational and security requirements when determining the appropriate symmetric encryption key strength for use.

- **Key-validity**
  - In this policy, enterprises must determine how long a specific symmetric encryption key can be used by applications. The OASIS SKMS BSA supports the specification of this policy in one of three methods:
    - **Date-based**: In this method, each symmetric key is valid for use from whenever the key was issued until a specific date and time. There are no limitations on the number of objects that can be encrypted with this key by authorized applications, until the specified date and time. At that time, all conforming implementations must stop using the key for encryption, but may continue to use the key for decryption if the key is accessible to authorized applications;
    - **Duration-based**: In this method, each symmetric key is valid for use from whenever a client starts using the key until the end of the specified duration period. There are no limitations on the number of objects that can be encrypted with this key by authorized applications until the specified duration has expired. At that time, all conforming implementations must stop using the key for encryption, but may continue to use the key for decryption if the key is accessible to authorized applications;
    - **Transaction-based**: In this method, each symmetric key is valid for a precise number of encryption transactions from whenever the key is first used by an authorized SKMS client. There is no date-time or duration based limitation on the key other than it cannot be used for greater than the specified number of encryption transactions. At that time, all conforming implementations must stop using the key for encryption, but may
continue to use the key for decryption if the key is accessible to authorized applications;

- **Key-use**
  - While there is no explicit policy that determines who, or which application, may use symmetric encryption keys, this can be implied by creating a Default KeyUsePolicy, or by assigning specific policies to individual SKMS clients and/or groups.

  A conforming implementation will use a specific KeyUsePolicy for a requester, if a valid KeyUsePolicy exists for that requester. If an individual KeyUsePolicy does not exist, then the conforming implementation will use a specific KeyUsePolicy that applies to a group that the requester belongs to within the SKMS. If a group KeyUsePolicy does not exist, then the conforming implementation will use the Default KeyUsePolicy to determine the Key-strength and Key-validity for that specific symmetric key.

  Conforming SKMS implementations will permit as many individual and group KeyUsePolicies as an enterprise deems necessary for its business.

- **Caching policy**
  - In this policy, enterprises must determine whether an SKMS client may cache symmetric encryption keys on its local storage device, and if so, how many keys it may cache and for how long. The OASIS SKMS BSA supports the specification of this policy with the following properties:
    - **Maximum New Keys**: With this property, an SKMS specifies precisely how many new (not used for any symmetric encryption transactions) symmetric encryption keys may be cached locally on the SKMS client. The value “0” (zero) indicates that new symmetric keys may not be cached locally on the SKMS client, and all new symmetric keys must, implicitly, come from an SKS server at runtime;
    - **Maximum New Days**: With this property, an SKMS specifies precisely how many days can an SKMS client cache new (not used for any symmetric encryption transactions) symmetric encryption keys locally. The value “0” (zero) indicates that new symmetric keys may not be cached locally on the SKMS client, and all new symmetric keys must, implicitly, come from an SKS server at runtime;
    - **Maximum Used Keys**: With this property, an SKMS specifies precisely how many used (used for at least one symmetric encryption transaction) symmetric encryption keys may be cached locally on the SKMS client. The value “0” (zero) indicates that used symmetric keys may not be cached locally on the SKMS client, and all used symmetric keys must, implicitly, come from an SKS server at runtime;
    - **Maximum Used Days**: With this property, an SKMS specifies precisely how many days can an SKMS client cache used symmetric encryption keys locally. The value “0” (zero) indicates that used symmetric keys may not be cached locally on the SKMS client, and all used symmetric keys must, implicitly, come from an SKS server at runtime;
Use First: With this property, an SKMS specifies whether a conforming SKMS client uses its local cache to locate symmetric encryption keys, or uses the SKS server on the network for getting the symmetric encryption key. The value of “Network” implies that regardless of other values in the KeyCachePolicy, the SKMS client will always use the network-based SKS server to request a symmetric key – new or used.

With a combination of values within the KeyUse and KeyCache policies, an SKMS implementation can exercise enormous flexibility over how symmetric encryption keys are used in the enterprise. Ultimately, the enterprise’s business, operational and security requirements must dictate the specific values for these policies.

3.4. Security Considerations

When an SKMS is implemented and deployed in an enterprise, it will become one of the most important security infrastructures of the company. The encryption keys managed by the SKMS will be protecting data across the enterprise, and as a consequence, the security controls around the SKMS – and the PKI it depends on - will be paramount.

The OASIS EKMI TC recommends the following security controls for an SKMS infrastructure, in all environments except for Development and/or QA environments:

a) Separate the SKMS servers, physically, from all other servers in the Data Center by keeping them in a separate locked cabinet whose keys are under the control of as few people as possible;

b) Do not permit any single individual to gain physical access to the SKMS servers; ensure that multiple people – a minimum of two people who do not work under the same manager – are required to be present to physically access the SKMS servers. Ideally, one of the individuals present should be from the Internal Audit group to document whatever activity is being performed;

c) Do not install any SKMS component without verifying the cryptographic message digest of the software; maintain this cryptographic message digest for audit purposes and as part of the SKMS’ integrity controls;

d) Do not install any other application – perhaps with the exception of PKI-related applications, such as a Certificate Authority (CA), Registration Authority (RA) or Validation Authority (VA) – on the same machine as the SKMS servers;

e) Do not install SKMS servers on virtual machines. Virtual Machines (VM) are relatively new technology for the UNIX, Linux and Windows environment, and as such, little research has been conducted into their safety;

f) Do not operate SKMS servers without cryptographic hardware security modules (HSM). The HSMs are the only assurance that a surreptitious attack has not copied away the Private Keys of the SKS servers, thus compromising the infrastructure;

g) When using HSM’s ensure that the HSM supports M of N access control. This is an implementation of technology to ensure that at least M of a pool of N authorized human entities are present with authenticated credentials before the HSM can be operated;

h) Ensure that all SKMS servers have locked down their hosts by minimizing the number of services running on the machine(s) and by implementing firewalls on them to block all ports other than those necessary to operate the SKMS;

i) If the database and the application servers of the SKMS servers must be
separated for any reason, ensure that each of the machines follows the same security controls;

j) Since all database records are encrypted and digitally signed, it is acceptable to maintain the database backups without any additional security controls around them;

k) Do not allow SKMS Administrators, Security Officers and/or Auditors to access the SKMS Administration Console without the use of two-factor authentication with strong asymmetric-key cryptographic authentication between the client and server;

l) Use only a CLOSED PKI for issuing SKMS credentials to clients and servers. A Closed PKI is one in which the enterprise has established a self-signed Root Certificate Authority and is in full control of all certificate-issuances under this Root CA. This control ensures that there is little risk of a public CA issuing a certificate to a third-party that may be able to connect to the enterprise's SKMS and request key-management services. Note that this control does not prevent an enterprise from contracting with specialized service providers to operate the Closed PKI on its behalf, as long as the PKI chains to a self-signed Root CA under the control of the enterprise;

m) It is very strongly recommended that enterprises use cryptographic HSM's on SKMS clients to protect the Private Keys of the clients' credentials. Trusted Platform Modules (TPM) on the motherboard of computers, smartcards, PCI-based cards, etc. are some examples of HSMs that can be used on the client. Without the use of an HSM, it is very difficult for an enterprise to prove that it is in full control of the cryptographic keys that are requested by a specific SKMS client. However, the implementing enterprise must make a determination on the appropriate cryptographic module for their clients based on business, operational and security requirements;

n) Do not operate an SKMS without using the Network Time Protocol (NTP) on all SKMS clients and servers. It is almost impossible to control SKMS policies unless all entities in the SKMS conform to using the same date and time values across the enterprise;

3.5. Technology

3.5.1. Symmetric Key Services Server

3.5.2. Symmetric Key Client Library

4. Hardware Security Module(s)
4.1. Standards and Protocols
4.2. Operations

5. The future of EKMI
  5.1. Short-term
  5.2. Long-term

6. EKMI Resources
  6.1. PKI Resources
  6.2. SKMS Resources

7. Appendices
Introduction

Components of an EKMI

PKI

What is a PKI

PKI, Public Key Infrastructure, is the gathering name for a range of standards, with the ISO X.509 standard in the middle. The basic purpose of a PKI is to issue digital certificates to entities. The digital certificates, as being infrastructure items are then used in a variety of applications for a broad range of purposes.

PKI is currently, and has been for several years, the dominant standard when it comes to strong authentication, digital signatures and communication encryption. The most pervasive example is that all web servers providing encrypted pages using the SSL protocol must have a certificate issued from a PKI.

Since PKI is an infrastructure component, in itself does not have any good usage, PKI is an enabling technology, enabling other application to be secure and thus available for purposes not possible without strong security. The PKI infrastructure is used to provide several security services to application some of them being:

- Authentication
- Integrity and non-repudiation, i.e. digital signatures
- Confidentiality, i.e. encryption

Most applications combine several services and some examples of real-world applications are:

- Web servers providing encrypted services, i.e. server side SSL.
- Web servers requiring strong client authentication, i.e. dual authenticated SSL.
- Secure email.
- Digital signatures on documents, for example PDF documents.
- Smart card login to workstations.

The architecture of a PKI consists of several entities where the basic ones are:

- Certification Authorities (CA)
- Registration Authorities (RA)
- Repositories (directories)
- End entities

To have a functional PKI at least a CA and a RA is needed. The two can be combined in a single application to make the implementation easier, or they can be separate, depending on the requirements on the PKI.

Rfc 4210 describes all the PKI entities and their relationships in a very complex picture. The good news is that much of the complexity can be hidden if the requirements on the PKI are less general and more specific, as in the case of providing security services for an EKMI. Off-course there is nothing preventing you from using an existing, or setting up a new, enterprise PKI to encompass all the needs of an enterprise, including EKMI.
In a PKI CAs form a hierarchical structure:

![Diagram of PKI hierarchy]

The hierarchy can be as small, only one CA, or as large as you like. The structure depends on the enterprise organization, policies and many factors.

Why PKI?

PKI

Applications

In this document a EKMI uses PKI to provide encryption of the communications and strong authentication between server and client. Both the server and client need to be certain that it is communicating with a valid entity, and needs to know the identify of that entity in order to make decisions.

The application of PKI in EKMI is:

- **Strong authentication of entities.**
- **Encryption of messages containing encryption keys**

Architecture

To run a PKI for EKMI a simple CA hierarchy and one or several RAs are needed.

Generally for an EKMI it is sufficient with one CA issuing certificates to all EKMI clients and Key Management Servers.
Handling the other aspects of the complete system is best accomplished by adding additional CAs.

- One Root Certificate Authority
- One Administration Certificate Authority
- One EKMI Certificate Authority

The administration CA will issue certificates for administrators and the EKMI CA will issue certificates for the EKMI entities.
Policy Considerations

It is commonly recommended that a PKI issue a Certificate Policy (CP) and a Certification Practices Statement (CPS) describing the rules and regulations for the applicability of certificates issued from the PKI. According to X.509, a certificate policy (CP) is "a named set of rules that indicates the applicability of a certificate to a particular community and/or class of applications with common security requirements."

IETF RFC 3647 provides a framework for CP and CPS writers. Following the framework and structure of the rfc makes it possible to compare different policies and a manageable way. Therefore is is highly recommended to use the outline provided in RFC 3647.

Some considerations when writing a CP/CPS concerns areas such as:

- Whether a hardware security module is required.
- Requirements for logging and audit.
- Separation of duties for different functions and staff.
- Requirement for audit and training of personnel.
- Whether dual control is required for certain functions, for example backup and restore of CA signature keys or physical access to servers.
- Availability of services, and allowed down-time in case of disaster.
- Identification of entities that will be issued certificates.
- Publication of revocation information.
- Compliance and other assessments.
- Representations and warranties.

There are many factors to consider when setting up a PKI. As long as the cost and benefit of different actions are weight in, the work of creating the CP/CPS should not be over-stated.

Unless your organization is very large with tight separation of duties between departments describing the requirements and departments implementing them, it is a good idea to combine both the CP and CPS in a single document. This makes it much easier to both write and read this documentation.

There are a few existing standards that can have effect on your policy. In Europe there are ETSI requirement for qualified certificates that mandates both administrative and technical issues. In an EKMI perspective the ETSI requirements will not affect the policy though, since there should be no need for qualified certificates in an EKMI.

If an EKMI is used to protect financial transactions, the PCI Data Security Standards (PCI DSS) may be relevant. Especially the PCI DSS Security Audit Procedures will be relevant by implicitly defining a policy requirements.

Best Practice Policies for PKI

The EKMI TC have developed Policy Guidelines with recommendations deemed useful for optimal EKMI implementations. These guidelines should be incorporated in your PKI policy.

Security Considerations

The security of the EKMI largely depends on the security of the PKI. If the PKI is compromised, the EKMI and all the encryption keys contained within should be considered suspect of being compromised as well.

Security considerations for a PKI contains basically the same security considerations as for other mission critical systems, with some additional elements.

- Hardware security modules, what level of protection is needed for CA signing keys?
● Network security, should the PKI be exposed to the Internet at all?
● Host security, patching, monitoring, backup etc.

In addition to the technical security considerations there are some additional procedural consideration to make:
● Privacy, should certificates be published? What can the Distinguished Names contain if the certificates are publicly available?
● Subscriber requirements. What kind of security measures can we require from subscribers? This can differ if they are end users or other computer systems?
● Who can register new EKMI clients and how are EKMI administrators authenticated and authorized in the system?

There are many more security considerations to think of. Following the RFC 3647 policy outline is a good help to make sure most of them are covered.

The PKI should be treated, security wise, as one of your most critical IT systems.

It should be possible so separate functions in the PKI, so that the CA can be isolated with all incoming communication going to the RA.

Security considerations for the CA:
● Strong protection of CA signing keys
● Not generally accessible on the network
● Strong authentication and separation of duties for administrators

Security considerations for the RA:
● Strong authentication of RA personnel
● Enrollment using credentials that can not be re-used

Technology

The technology part of these guidelines will cover the basic PKI components and provide guidelines for their configuration in an EKMI environment. It is important to remember that a PKI can be set up and configured in many ways, from a large national infrastructure to a small personal SSL CA. These guidelines will focus on the needs of an EKMI, and not take into consideration that you may be willing to set up a global PKI for a large enterprise, and use that also for EKMI. These guidelines may be worth reading also of you intend to set up a global PKI, but there will be additional issues to wrestle with in such a case.

Certification Authority

A Certification Authority, CA, is the component that issues certificates. The CA is the trust anchor in the system and signs issued certificates. By doing this all issued certificates can be trusted by verifying signatures up to the trusted CA. The CA can also revoke certificates if the private key has been compromised or the entity has gone out of operation, or for some other reason is not trusted anymore.
Specifically the EKMI CA will not issue a large number of certificates for usage by end users, i.e. for email signing, web authentication, document signing etc. Therefore the EKMI CA will issue a limited number of certificates compared to the more general PKI case.

Certificates used in the EKMI will be used to authenticate and protect web service messages.

When a new application server is introduced in the system it will be configured with a private key and a certificate. The enrollment process used depends on the EKMI client library used, and can be manual or semi-automatic. A trusted administrator will register the application server and approve the issuance of certificates. This process should be as easy, but still secure, as possible and the process should be adapted to the organizations work-flow.

Registration Authority

A Registration Authority, RA, is where administrators register new entities. Entities can be users or servers, in an EKMI it will often be servers in need of symmetric encryption keys.

Before the CA issues a certificate the entity must be registered. Exactly how this is done depends on the administrative work-flow chosen. The registration work-flow should be chosen carefully so it fits in the organizations and does not put unnecessary burden on the administrators.

Generally for an EKMI it is sufficient with one RA integrated on the same platform as the CA. The RA is remotely accessible by administrators, so it can still be used from different physical locations.

A typical work-flows is outlined below.

1. A trusted administrator registers a new application server in the RA. Here the data about the EKMI client, such as dns name, is entered.
2. A username and a one-time password is generated.
3. The username and password is sent to the application server administrator, who performs the enrollment on the EKMI client side.
4. The application server administrator performs the enrollment process using the username and password.
5. The EKMI clients private key and certificate is stored on the application server.

The registration process is a process performed once for every client introduced in the EKMI. Certificates however have a limited validity, so they must be updated. How often this happens depends on the organizations policy, but usually the validity is in the range of 3-10 years. Three years is usually a good compromise between maintenance and security. If the organization have a good process for revoking compromised keys, or use hardware security modules on all clients, a process with automatic updates of expiring certificates can be put in use.

Personnel using the RA functions are usually spread out through the organization.

Distributed RA operations for efficient usage across the enterprise
When it comes to the RA it will be a core work-flow for your administrative personnel.

An RA must be flexible and support integration and customization through different protocol and APIs. Efficient large-scale PKI implementations have customized RA functions that integrate will in the organization.

Standards and standard protocols is a key issue for any interoperable PKI.

The RA must support X.509 and IETF standards and protocols.

**Hardware Security Module(s)**

In a PKI asymmetric encryption techniques are used. In asymmetric encryption there is one private key and one public key. The private key must be kept secret, and the public key can be shared with all other users. One way to store keys are in regular files on a hard disc, which is very efficient and low cost. The drawback with such storage is that the file can easily be copied by an intruder and used on any other system, impersonating the original system.

Hardware Security Modules, HSMs, are used to prevent copying if private keys. A HSM is a device specially designed to keep private keys impossible to copy out from the HSM. The HSM will perform all operations involving the private key and usually requires a password to activate the key. This makes it impossible to copy the private key by other means than to steal the HSM itself along with the activation password. Using a HSM also makes it easy to de-activate private keys by simply stopping the HSM.

HSMs comes in many forms:

- Network attached devices, usually deployed in a server room, where several trusted servers can use the HSM resource.
- Local PCI cards connected inside a server.
- Smart cards, credit card size chip cards.

Which one to use depends among other things on the speed requirements (smart cards are much slower) and budget (smart cards are much cheaper).

In a typical EKMI you may use:

- A network attached HSM for the CA and KMS. If located in the same server location they can share one unit.
- Local PCI cards or smart cards on EKMI clients, depending on the usage pattern.

You should use an HSM that allows you to deploy the EKMI to the highest required security standards, with K/N operators needed to perform certain operations.

**Publishing Directory**

Using a publishing directory means storing issued certificates in an LDAP directory. The directory is used to search for the certificates of other entities to be able to verify data from, or encrypt data to those entities.

A directory is not necessary in an EKMI.
A directory is not structurally necessary in an EKMI, since the EKMI clients and the KMS will know about each other in advance. Depending on the software implementation a directory may be needed for the operations of the system. The characteristics of the directory server is dictated by the EKMI software.

**Certificate Validation**

Certificate Validation means the process of verifying that a certificate is valid. If a certificate is valid the authentication and authorization process can continue, but if a certificate is invalid the process will stop and the client will be denied.

The process of validation includes the following checks:

- that the certificate is not expired
- the signature on the certificate verifies up to the trust anchor, i.e. the CA certificate
- that the certificate has not been revoked
- that the key usage declared for the certificate is consistent with the actual intended usage, i.e. authentication

The KMS performs certificate validation on all clients that makes requests to the KMS. After passing certificate validation the KMS moves on to authorization and policy decisions regarding the EKMI.

For EKMI administrators especially revocation information and certificate expiration are important. Administrators must take steps to ensure that clients taken out of order, or stolen, are revoked. They must also make sure that certificates that are about to expire are updated. If expiring certificates are not updated, the clients will not be able to contact the KMS, and the system will not work.

**Use on-line OCSP service for verifying revocation status of entity certificates on the KMS.**

Even though the OCSP protocol is on-line, the updating of the OCSP-responder can still be based on CRLs, therefore introducing a delay between revocation and when the OCSP-responder is updated.

**The OCSP-responder should be updated with revocation information in real-time.**

**References**

IETF RFC 3280, Certificate and Certificate Revocation List (CRL) Profile
IETF RFC 2560, Online Certificate Status Protocol – OCSP
IETF RFC 4210, Certificate Management Protocol (CMP)