Johan Ivarsson, johan@certezza.net
Andreas Nilsson, andreas@certezza.net
Certezza AB
Stockholm 2010-12-31

A Review of Hardware Security Modules
Fall 2010

Table of Contents

Foreword

This review was performed on commission from .SE (The Internet Infrastructure Foundation), whose influence ends there and Certezza's independence is thereby intact.

The primary goal of the review is to evaluate commercially available Hardware Security Modules (HSMs) from a technical perspective. In addition an effort has been made to describe the respective vendors' solutions in commonly used, standardized terms. The primary focus has been on HSM usage in larger DNSSEC deployments. However most of the findings are application agnostic and as such the review can be applied to HSMs in general. Lastly, the review aims to encourage vendors to continuously improve their products.

Certezza is an independent information and IT security company offering solutions for secure IT infrastructures. Certezza has extensive experience in carrying out analyses, reviews and preliminary studies, combining a structural approach with expertise in the sector of information and IT security.

Abstract

This report describes a technical review of four leading network based Hardware Security Modules performed during the fall of 2010. When deriving the review point set the focus was primarily on security features and functionality used for DNSSEC applications. However the more interesting findings were in different areas such as usability and management procedures.

Generally all the modules work as expected and offer the necessary functionality one needs from a secure crypto processor. Which HSM to choose depends on budget, the deployment scenario, performance requirements and other application specific facts. From an application perspective the PKCS#11 interface worked exemplary on all modules. Once set up we hardly experienced any problems with the interface. The only issue worth mentioning is the fact that we needed to execute several concurrent threads (for all modules) in order to achieve a decent HSM CPU load.

There was high level of diversity in how features such as role structure, authorization models and key backup were implemented. A more standardized security and authorization model and nomenclature would have been favourable. Instead each vendor has chosen to integrate with the PKCS#11model in different fashions. An evolvement of the PKCS#11 standard to incorporate more complex than smartcards would probably be advisable.

When performing this review it would have been very helpful to have had access to best practise recommendations for setting up and configuring HSMs. Such a text could also document certain application areas and general deployment scenarios. At the moment the user is referred primarily to vendor specific whitepapers and presentations.

1 Introduction

1.1 Background

As a first time HSM user or buyer it is not straightforward to know where to begin. This study aims to bridge the gap between product whitepapers and actually using the appliances in practise. We look at how different vendors implement crucial features such as secure key storage, backup, recovery and HA clustering. The aim is not to perform a proper product comparison but rather to identify important HSM features and look at which different strategies have been used to implement these features in practise.

1.2 Technical overview

A Hardware Security Module (HSM) is a secure crypto processor with the main purpose of managing cryptographic keys and offer accelerated cryptographic operations using such keys. The modules typically offer protection features like strong authentication and physical tamper resistance. Main features of an HSM include on board key generation and storage, accelerated symmetric and asymmetric encryption and backup of sensitive material in encrypted form.

HSM systems come in different flavours and form factors. Common HSM types are smartcards, PCI plugin cards and full-fledged, physically shielded LAN-based appliances with features like thermostats to detect tamper attempts. In this study we have chosen to look at the last type. The two main advantages of the network attached HSM types (compared to PCI-based) are that they are inherently platform independent and can be used simultaneously from several clients. Smartcards are not appropriate for application areas with any type of higher performance requirements.

Traditionally HSMs have been used in the banking sector to secure large amounts of bulk transactions. Other common usage areas are to secure CA keys in PKI deployments and SSL acceleration. In the last couple of years with the advent of DNSSEC (DNS Security Extensions) an increased focus has been placed on storing DNSSEC keys, and encrypting zone records using an HSM.

1.3 Purpose

The purpose of the review is to test the HSM from both the viewpoint of the application and the system administrator in order to assists implementers and encourage product improvements. The main focus when choosing the review points has been on lager DNSSEC deployments. A secondary purpose has been to describe the respective vendors' solutions in commonly used, standardized terms. The HSM knowledge area, especially with regards to authorization structures and notions is surprisingly diversified.

1.4 Scope

The primary scope of the study has been PKCS#11 usage of network attached Hardware Security Modules. The following products and topics have been left out of the study.

- No pure PCI card HSMs integrated directly on a client platform have been used.
- No tests have been performed to verify the physical security measures implemented on the modules. Instead we choose to rely on the respective FIPS 140-2 validations.
- No practical test of application interfaces except PKCS#11 have been performed.
- Even though some cursory testing has been done on symmetric algorithms the main focus has been on asymmetric algorithms, and especially RSA.
- Some HSMs have dedicated PKI functionality. This has not been tested in practise.

2 Review method

The review of the products was performed in Certezza's office according to the test review points listed in chapter 4. Each of the review points was evaluated separately according to the laid out test plan. The claimed functionality for each of the tested products was verified and documented.

2.1 Selection criteria

The vendors were chosen based on two main criteria; high deployment rate at customer sites and/or a good reputation as an HSM vendor.

The actual product selection was based on the following requirements:

- Network connected appliance
- FIPS 140-2 Level 3 validated as described in [FIPS140-2].
- Signing performance: preferably at least 1500 sign/sec for 1024-bit RSA keys.
- Automatic synchronization of keys between HSM systems.
- Ubuntu support for the PKCS#11 library.
- Support for separation of duties/division of command.

2.2 HSM selection

The products were chosen according to the selection criteria in section 2.1.

Vendor	**Model reviewed**	**Form factor**
AEP	Keyper v2	
Safenet	Luna SA 4.4 (PED)	
Thales	nShield Connect 6000	
Utimaco	CryptoServer Se1000	

Note1: Most of the vendors above have a range of different models in different price ranges. We intended to include list prices in the review as well. However since most deployments are unique with regards to redundancy and backup requirements and different vendors solve these issues differently the list price for a single HSM is not always a good measure. PCI-based HSMs are naturally cheaper than their LAN-based counterparts.

Note 2: Even though ARX PrivateServer HSM did indeed fulfil the requirements in section 2.1 we did not receive any reply from the vendor regarding participation in the study.

Note 3: All FIPS-validated cryptographic modules can be found in a list published by NIST at [FIPSVal].

2.3 Test setup

In order to minimize performance impact from network latency the lab setup was very simple consisting of:

- A single Ubuntu 8.04 client running the test applications. When possible we used the Ubuntu client for administration as well.
- The four network attached HSMs located on a dedicated LAN directly connected to the application client.

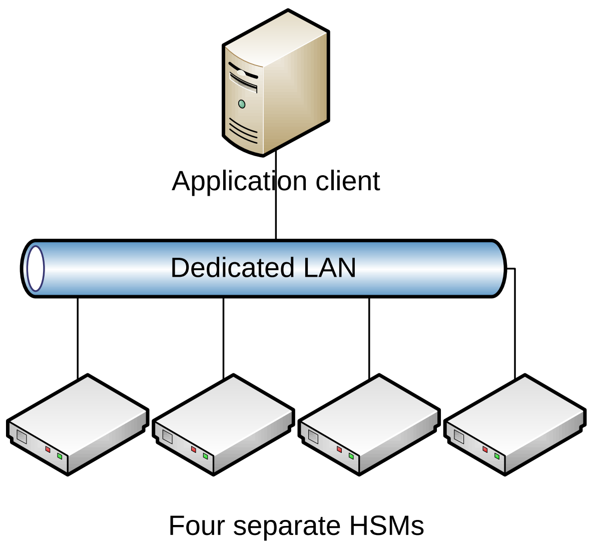

Figure 1 – Schematic view of the network setup in the test lab.

2.4 Test tools

We used the management software and PKCS#11 software libraries distributed with the HSMs.

For the performance testing we used the command line tool ods-hsmspeed. The tool is bundled with the open source DNSSEC management system OpenDNSSEC [OpenDNSSEC].

For the test of the PKCS#11 interface (review point A.2) we used a specially developed test tool called pkcs11-testing. If desired, please contact the authors to obtain the source code.

3 Security Models

This chapter is a complement to the security features discussed in section 4.3 and summarizes the security and usage model for each HSM. Since there is no standardized nomenclature for HSM concepts it is necessary to explain each vendor's notions as a background to the text in the review findings.

The security features and notions presented in section 3.1 are common between all the tested HSMs.

3.1 Common features

There are a number of security architecture decisions that are shared between the four HSMs.

- As opposed to many other similar appliances, the modules expose no Web GUIs. Administration is done primarily on the physical front panel.
- Remote administration is done via client software over some form of secured CLI shell.
- Smart card authentication is used to authorize security related operations. Division of command in an m-of-n fashion can be applied if desired.
- PKCS#11 interface over a normal TCP/IP routable Ethernet connection. The keys generated and used within the PKCS#11 contexts are normally called application keys. It is conceptually important to separate these keys from keys used for HSM management as described in the security models.
- No sensitive information such as cryptographic key material ever leaves the HSM unencrypted.
- All evaluated HSMs are at least FIPS 140-2 Level 3 validated. As part of the validation process it is mandatory to supply a security policy which details the security architecture especially with regards to key management. The respective security policies are linked from NISTs list of FIPS 140-2 evaluated cryptographic modules [FIPSVal].
- Except for the AEP Keyper, the actual HSM functionality is placed on an integrated PCI card which can also be bought separately and installed directly on a client machine. The FIPS 140-2 validations apply to the cryptographic functionality of the modules. Therefore it is relevant to analyse network communication protocols and authentication mechanisms even though the HSM is FIPS 140-2 validated.

3.2 PKCS#11

3.2.1 Overview

The standard PKCS#11 (Public-Key Cryptography Standard) defines a platform agnostic API to cryptographic tokens [PKCS#11]. PKCS#11 was initially designed for accessing smartcards and includes notions such as slots (smartcard readers) and tokens (smartcards) which imply smartcard technology. Inherent in the design is the ability to use several tokens concurrently through the same API.

PKCS#11 defines two roles, crypto officer and user, with separate PINs. The crypto officer PIN is used to manage the user role. The user PIN is used to authorize token usage operations.

3.2.2 PKCS#11 and HSM technology

Even though HSMs have evolved to incorporate more complex devices with advanced authentication and authorization models PKCS#11 is still the de facto standard for platform agnostic applications. The cryptographic operations are the same over different HSM types but the administration structure and authorization models vary.

Use of PKCS#11 notions for complex HSMs is sometimes a bit confusing. There is no logical mapping of either a token (smartcard) or a slot (smartcard reader) to PCI-cards or a network attached HSM appliance. The vendors have solved this inherent conflict by letting the HSM administrator divide the HSM into logical tokens.

The PKCS#11 user role translates fairly well to the application calling the library. When the token is set up the HSM administrator (PKCS#11 crypto officer) sets the user PIN on the token. This PIN is used by the application when accessing HSM functionality. The PKCS#11 crypto officer role is however not used in its pure form for complex HSMs. The authentication logic for the crypto officer is not part of the virtual token as for smartcards but lies in the larger scope of HSM administration. The crypto officer is typically authenticated using some form of strong authentication, normally a smartcard which complicates the situation further.

In the security model for each HSM a short summary is given of how the HSM is divided into logical tokens and which roles correspond to PKCS#11 crypto officers and users.

It should be noted that PKCS#11 does not automatically name keys in a fashion appropriate for backup and recovery. It is the responsibility of the application to name the keys adequately upon creation.

3.3 AEP Keyper

3.3.1 Overview

As mentioned in the previous section the AEP Keyper is the only appliance in this test which is not based on an integrated PCI card. It is also the only HSM which is FIPS 140-2 Level 4 validated.
Its foremost design objectives are to be secure and reliable (which can be noted in fewer supported network appliance functions and slower performance). On the other hand the security model is very clear and easy to follow.

3.3.2 Key and role structure

A the core of the AEP Keyper is the Storage Master Key (SMK) which is used to encrypt all application keys on board the HSM. All tampers with the modules result in positive destruction of the working storage and the SMK.

The other important key is the Adaptor Authorization Key (AAK). The AAK is used to protect the smartcards of the Security Officer and operator (the two administrative roles on the Keyper). In order for two Keypers to be administrated by the same set of Security Officer smartcards the modules must share AAK.

There must be at least two Security Officers in an m-of-n command structure. The Security Officer role is authorized to perform all HSM and key management tasks.

The sole responsibility of the Operator is to set the Keyper online/offline, in other words activate its network interface.

Other roles that are not directly tied to operational responsibilities are Key component holders (holds a smartcard with a part of a SMK or AAK backup) and Application Key Holders (ditto for application keys).

3.3.3 Key storage

All keys are stored in NVRAM on the module or on dedicated backup tokens.

3.3.4 PKCS#11

Tokens are created by the Security Officers which hence represent the PKCS#11 crypto officers. A user PIN is set by the Security Officers on the PKCS#11 token for application use.

3.3.5 Communication

The AEP Keyper is designed for use on private, secured networks. It would be possible to front one or more Keypers on a private network with an AEP load balancer that would act as a bridge between the Keyper and a public network.

3.4 Safenet Luna

3.4.1 Overview

First of all it should be noted that the Safenet Luna can be ordered in two different modes; password based authentication (FIPS 140-2 Level 2) and Trusted Path authentication (FIPS 140-2 Level 3). For this review we have focused entirely on option two, which is based on command authorization with so called PED (PIN Entry Device) keys. The PED keys are essentially smartcards which are used together with a supplied numerical keyboard called a PED.

The Luna HSM is separated into partitions. A partition is a dedicated part of the appliance's NVRAM with separate management. It is possible for a single partition user to own and manage several partitions, see role 3 below. One or several clients must be assigned to a partition before it can be used by applications.

3.4.2 Role structure

There are three main administrative roles involved in Luna HSM management.

1. The appliance administrator logins via SSH or serial terminal. The appliance administrator can perform general, appliance-level administration. Additional authentication is required to perform HSM administration tasks and partition management, see below.
2. The HSM administrator is authenticated using a blue PED key. The HSM administrator is authorized to edit HSM-wide policies, backup and restore the HSM, create and remove partitions.
3. The Partition user (Crypto Officer) is authenticated using a black PED key and is authorized to set partition policies, assign partitions to specific clients etc.

If m-of-n is used for the HSM administrator or the partition user additional green PED keys are required for authentication.

There are five types (colours) of PED keys:

1. Blue – HSM administrator PED key
2. Black – Partition user PED key.
3. Red – Key cloning PED key which is used to synchronize key material among multiple modules in a common "domain". The red key verifies that the hardware is under control of the own organization. The main purpose is for clustering.
4. Green – Used to implement m-of-n division of command.
5. Orange – Used for remote administration via a so called Remote PED.

3.4.3 Key storage

All keys are stored in hardware on the module or on dedicated backup tokens.

3.4.4 PKCS#11

There is a one to one mapping between a Luna partition and a PKCS#11 token. By default the partition user role represents a crypto officer, but the black PED key can also be extended to include a second pure user PIN.

3.4.5 Communication

Once setup the Luna is designed to be used over public networks. Safenet implements an SSL-like protocol called NTL (Network Trust Link) to encrypt the traffic. The session key is derived using initially exchange server and client certificates.

3.5 Thales nShield

3.5.1 Overview

The basis of the Thales nShield security architecture is the Security World concept. A Security World can comprise several HSMs but is administered using a single set of administrators.

Another important notion is the Remote File System (RFS). As the name indicates the RFS is not stored on the HSM but on a client machine, not necessarily the same as is used to administer the appliance or run the PKCS#11 library. The RFS contains keys, configuration data and optionally log files synchronized from the HSM.

3.5.2 Role structure

The Security World is administered by an Administrator Card Set (ACS) consisting of n number of smartcards of which m are required to authorize administrative tasks. The ACS is used to encrypt Security World recovery data and authorize creation of Operator Card Sets (OCS).

A Security World contains a single ACS (possibly backed up), but can contain several Operator Card Sets (OCS) with m-of-n structure. An OCS is used to create and encrypt application keys. Both ACS and OCS consist of smartcards. It is also possible to use so called softcards to encrypt application keys. Softcards are essentially soft certificates with a password protected private key.

With regards to form factor, there is an integrated smartcard reader on the front panel. For remote administration possibilities see evaluation point C.4.

3.5.3 Key storage

The keys are stored in the RFS, encrypted using an ACS, OCS or a softcard.

3.5.4 PKCS#11

There is a direct one to one mapping between an OCS and a PKCS#11 token. The PKCS#11 token label will automatically be set to the name of the OCS. The OCS card holders vaguely maps to the PKCS#11 crypto officer role since the OCS is used to manage the tokens.

3.5.5 Special features

The nShield architectures provides a specialized cryptographic interface called CHIL (Cryptographic Hardware Interface Library) described as "a simple programming interface for accelerating modulo exponentiation and accessing the RSA/DSA keys that are used by some application software."

An extra feature is the ability to write "CodeSafe applications" in Java or C/C++ which can be executed directly on the HSM using a so called Secure Execution Engine (SEE).

3.5.6 Communication

The communication between the client and the RFS is encrypted using keys exchanged when setting up the RFS initially.

3.6 Utimaco CryptoServer

3.6.1 Overview

As opposed to the other modules the CryptoServer is delivered with a default admin account. The credentials of the default administrator should be changed or the entire account replaced before the HSM is used in a production environment.

3.6.2 Key structure

At the core of the CryptoServer is the Internal Master Key which is used to encrypt internal sensitive material on the HSM. The other important key is the Master Backup Key (MBK) which is used to protect back up data. The MBK is stored internally and/or split into two or more smartcards.

3.6.3 Permission structure

There are no predefined administrator roles and no traditional implementation of division of command. Instead all users are considered equal in type but can contribute differently to the total authentication state of the HSM.

The authentication state consists of eight values in the range [0,3] with 3 being the highest possible authentication level. An example of authentication state could for instance be

[2,0,0,1,0,3,0,1]

Each user has a corresponding eight-value array. The above state could have been achieved by two concurrently logged in users having permission arrays [2,0,0,0,0,0,0,0] and [0,0,0,1,0,3,0,1]. The two first slots represent user management and CryptoServer administration and the default admin account has permission array [2,0,0,0,0,0,0,0].

The permission model is very flexible but (in our opinion) probably unnecessarily complex for most deployments. There is also a risk in leaving the decision to remove the default admin account up to the customer/user.

3.6.4 Key storage

When using PKCS#11 all keys are stored in hardware on the module or on dedicated backup tokens. For other interfaces (CXI, CAPI or CNG) it is configurable if keys are stored on board the HSM or in software on the client.

3.6.5 PKCS#11

Tokens are created by any administrator with high enough privileges. By default a crypto officer PIN and a user PIN must be set for the token. It is also possible to use Secure Messaging (see 3.6.5) for PKCS#11 sessions. In that case, a special PKCS#11 session user is created which can use any available authentication method listed in C.2.

3.6.6 Communication

The CryptoServer can be configured to be used on public networks. Administration commands and PKCS#11 communication between the client and HSM can optionally be encrypted which is called “Secure messaging”. The session key is established either using Diffie Hellman key agreement or through user authentication.

4 Review results

This section lists the test reviews points and the corresponding vendor support. Note that we are not recommending one vendor (or setup type for that matter) over another but simply trying to evaluate how well certain functions are implemented. It should be noted that all the modules were tested while running in FIPS mode. Some of the modules disable certain key lengths and cryptographic functionality in FIPS mode.

4.1 Technical data (A)

A.1 Algorithm support – Support for algorithms and key lengths commonly used for DNSSEC and PKI applications. Most of the modules support additional cryptographic features. For a complete list of cryptographic PKCS#11 mechanisms per HSM see Appendix B.

Function	AEP Keyper	Safenet Luna	Thales nShield	Utimaco CryptoServer
AES*	128-256	128-256	128-256	128-256
AES modes	ECB,CBC,MAC	ECB,CBC,MAC	ECB,CBC,MAC	ECB,CBC,CTR,MAC
3DES modes	ECB,CBC,MAC	ECB,CBC,MAC	ECB,CBC,MAC	ECB,CBC,MAC
RSA	1024-4096	512-4096	1024-4096	512-8192
ECDSA	-	112-571	-	112-521
DSA*	512-4096	512-1024	512-2048	512-4096
ECDH*	-	112-571	-	112-521
SHA-1	SHA1	SHA1	SHA1	SHA1
SHA-2*	SHA256-SHA512	SHA256-SHA512	SHA256-SHA512	SHA256-SHA512
HMAC	SHA1	SHA1-SHA512, MD5	MD5	SHA1-SHA512, MD5

*Part of NSA Suite B [NSASuiteB]

A.2 PKCS#11 interface – The most commonly used platform agnostic interface is PKCS#11. We created an automated script to test some main PKCS#11 functions to verify the library implementation. The following functionality was tested:

1. Initialization – Initialize library and set up session
2. Digesting – MD5, SHA1, SHA256, SHA512
3. RSA – key generation and signing
4. DSA – key generation and signing
5. ECDSA – key generation and signing
6. AES – key generation and encryption
7. Key agreement (ECDH) – key derivation
8. Public key information in private key object – The private key object potentially contains information about the public key as well, in which case only the private key needs to be stored. This is not guaranteed but optional in PKCS#11.
9. Import of an 1024-bit RSA key pair

Test #	AEP Keyper	Safenet Luna	Thales nShield	Utimaco CryptoServer*
1	√	√	√	√
2	√	√	√	√
3	√	√	√	√
4	√	√	√	√
5	-	√	-	√
6	√	√	√	√
7	-	Normal but not cofactor derivation	-	√
8	√	√	√	√
9	√	Not in FIPS mode**	Not in FIPS mode	√

* For some reason it was impossible to load other PKCS#11 libraries at the same time as the CryptoServer's.

** Support for importing (via wrapping) keys in both FIPS and non FIPS mode but not direct via PKCS#11 key import.

A.3 Supported interfaces (in addition to PKCS#11)

Module	Findings by the test team
AEP Keyper	MS CAPI, JCE, OpenSSL
Safenet Luna	MS CAPI and CNG, JCA/JCE, OpenSSL
Thales nShield	CHIL, MS CAPI, JCA/JCE, Check Point VPN-1/FW-1
Utimaco CryptoServer	MS CAPI and CNG, JCE, OpenSSL, CXI

A.4 Key storage capacity – If keys are stored on board the HSM the storage capacity is relevant.

Module	Findings by the test team
AEP Keyper	8000 1024-bit RSA keys
Safenet Luna	1200 2048-bit RSA keys The next generation Luna SA 5 will be able to store up to 20 000 keys.
Thales nShield	Very limited on board NVRAM storage, only recommended if legally required. Unlimited software key storage in the Security World/Remote File System.
Utimaco CryptoServer	5000 1024-bit key pairs per HSM, 1500 key pairs per PKCS#11 token.

A.5 Performance - For test lab setup see the description in section 2.3. Performance measurements are controversial and the results simply list the performance achieved in our test lab. In general our results are slightly lower but in the same range as the vendor provided figures.

All reviewed HSMs support both symmetric and asymmetric encryption. Our tests were performed for RSA signing as it is the most common HSM usage for DNSSEC and PKI applications. The testing tool used was ods-hsmspeed which is distributed with the open source DNSSEC signing software OpenDNSSEC [OpenDNSSEC].

In order to get decent performance from the HSMs, the test tool had to execute several independent threads creating PKCS#11 sessions. The results are therefore separated into two tables. The first table lists maximum performance when running multiple threads. The second table lists performance when running single threaded. To make the tests fair we experimented extensively to find the minimum number of required threads adapted to each separate HSM. The tests were executed long enough to minimize the overhead impact from setting up the sessions. The interested reader is referred to Appendix A for the number of threads and iterations per HSM.

Maximum performance (signatures/second)

Key size	AEP Keyper	Safenet Luna	Thales nShield	Utimaco CryptoServer*
1024	1020	7000	4375	2730
1536	580	1896	3560	1800
2048	410	1225	2760	1120
4096	23	45	410	260

* Running multithreaded tests did not improve performance for the CryptoServer. Instead the results in the table above were achieved by calling the library simultaneously from several processes. Running separate processes is not suitable for bulk signing applications such as DNSSEC. It was possible to do a temporary workaround by creating virtual slots in the configuration but it did not scale very well. An updated release of the PKCS#11 library with support for multithreading is scheduled for Q2 2011.

Single threaded performance (signatures/second)

Key size	AEP Keyper	Safenet Luna	Thales nShield	Utimaco CryptoServer
1024	310	800	950	1160
1536	140	570	740	920
2048	110	420	570	710
4096	13	35	150	230

It should be noted that in order to maximize performance it would be advisable to use a pure PCI card HSM instead of a LAN based to avoid the network overhead.

4.2 Communication and system (B)

B.1 Supported host operating systems

We used Ubuntu 8.0.4 on our common administration and application client.

Module	**Findings by the test team**
AEP Keyper	Most administration tasks are done directly on the front panel. There are tools for audit log extraction (Java-based) and firmware downloads (Windows 2003/2000/XP). The PKCS#11 library is available on Windows and *nix platforms.
Safenet Luna	Windows 2008/2003/2000 Solaris 9, 10 Linux RedHat Enterprise 4,5* AIX 5.3, HP-UX 11i *We had to do some manual package imports to make it work on Ubuntu 8.04.
Thales nShield	Windows 2008/2003/Vista/XP Linux, Solaris, HP-UX, AIX
Utimaco CryptoServer	Windows 2008/2003/ Vista/2000/XP) Linux, Solaris, AIX 5L

B.2 Backup functionality

Module	**Findings by the test team**
AEP Keyper	All key types as described in the security model can be backed up to dedicated smartcards.
Safenet Luna	Backup of HSM (excluding partition content) or individual partitions to specialized PCMCIA backup tokens. A release of a USB based HSMs called G5 is scheduled for Q1 2011, these modules can be used to back up the entire HSM.
Thales nShield	Backup is automatically handled through the use of a Remote File System (RFS) as described in the security model.
Utimaco CryptoServer	Cryptographic keys and the user database can be backed up, encrypted by the Master Backup Key (MBK). The MBK is created on the CryptoServer and can be stored onto two or several smartcards.

B.3 Synchronization and clustering – Possibility to share keys between HSMs in different operating locations to enable load sharing and hardware fault tolerance.

Module	**Findings by the test team**
AEP Keyper	Fault tolerance and load sharing via AEP load balancing software run on the client.
Safenet Luna	Possibility to create HA clusters consisting of several Luna modules the for load sharing and fault tolerance. A software client is used to virtualize the cluster into a single token for the application perspective.
Thales nShield	Load sharing is supported through the use sharing of keys between HSMs but it is up to the application to perform the actual load balancing.
Utimaco CryptoServer	Only possible if using the CXI interface, not for PKCS#11. Failover and load sharing can currently be implemented using PKCS#11 virtual slots, but not transparently for the application. This is scheduled for Q2 2011.

B.4 Network communication protocol between HSM and client

Module	Findings by the test team
AEP Keyper	All of the communication is sent over TCP/IP (routable) but unprotected. If the Keyper need to be used on public networks an AEP load balancer can be used as a proxy.
Safenet Luna	All of the communication is sent over TCP/IP (routable) and encrypted using the NTL (Network Trust Link) described in the security model.
Thales nShield	All of the communication is sent encrypted over TCP/IP (routable) using the Security World/RFS setup.
Utimaco CryptoServer	All of the communication is sent over TCP/IP (routable). Commands are optionally encrypted and signed using AES and AES MAC.

B.5 Multiple Ethernet interfaces

Module	Findings by the test team
AEP Keyper	No
Safenet Luna	Yes, two separate Ethernet ports.
Thales nShield	Yes, two separate Ethernet ports.
Utimaco CryptoServer	Yes, two separate Ethernet ports.

4.3 Security features (C)

For definition of vendor specific concepts and abbreviations, see corresponding security model in chapter 3.

C.1 Certification levels - The validations only covers the actual cryptographic chip, not the complete HSM including administration, backup etc. It is also worth to note that the certifications are firmware specific and the modules need to be re-certified for each new firmware version.

Module	Findings by the test team
AEP Keyper	FIPS 140-2 Level 4, FCC part 15 Class B, BSEN60950 Safety, BSEN61000 Susceptibility Performance B, BSEN55022 Level B Emissions
Safenet Luna	FIPS 140-2 Level 3 when using Trusted Path Authentication with PED keys (see security model) and Level 2 if using password authentication. CC EAL 4+
Thales nShield	FIPS 140-2 Level 3, CC EAL4+.
Utimaco CryptoServer	The model we tested, CryptoServer Se-Series LAN, is in the process of being FIPS 140-2 Level 3 validated. The CryptoServer CS-Series LAN is FIPS 140-2 Level 3 and ZKA (German Credit Association) validated".

C.2 Supported authentication methods – All of the modules use password protection on the PKCS#11 token for application usage.

Module	Findings by the test team
AEP Keyper	Smartcards for Security Officers and Operators. A minimum of two separate cards are required for security related operations.
Safenet Luna	Password or client certificates for appliance administrators. Password or PED keys for HSM admin and partition owners. A PED key is essentially an integrated smartcard used to authorize operations using the PED (Pin Entry Device).
Thales nShield	The Administrator and Operator Card Sets consists of smartcards. In addition softcards can be used for key wrapping.
Utimaco CryptoServer	All administrators/users can be authenticated using the following methods: Password (plain, SHA-1, HMAC) Client certificates stored on smartcard or in a key file. Direct smartcard logon onto the CryptoServer

C.3 Support for division of command (m-of-n or similar) – The concept m-of-n implies that m out of n designated administrators must be present to authorize a certain operation.

Module	Findings by the test team
AEP Keyper	Obligatory m-of-n with at least m = 2 for all roles.
Safenet Luna	Support for m-of-n for the HSM admin and partition owner roles. Extra PED keys (of green colour) are used in addition to the blue HSM admin or black partition PED keys.
Thales nShield	Support for m-of-n both for Administrator Card Sets (Security World Management) and for Operator Card Sets (PKCS#11 token handling).
Utimaco CryptoServer	The permission structure described in the security model allows independent m-of-n for all available commands.

C.4 Remote administration - Often times the HSM is located in remote or shielded locations. It might be inconvenient to gain physical access to these locations to perform administration tasks.

Module	Findings by the test team
AEP Keyper	Remote authentication is not possible. Unauthenticated operations such as audit log extraction are possible.
Safenet Luna	There are three options for remote administration: 1. Remote PED connected to client computer. 2. Use of a Remote Administration appliance to administer another HSM remotely. This option is only available when using Trusted Path Authentication (PED keys). 3. The password based solution (non-FIPS) is remote admin only since all authentication I performed via the SSH client.
Thales nShield	Remote administration is possible through the creation of a special Remote Operator Card Set.
Utimaco CryptoServer	The module can be administered remotely if using password or client certificate authentication methods.

C.5 Setup procedure

Note: Before beginning the actual setup it is important to make decisions about m-of-n administration, use of FIPS mode, backup functionality and other deployment regulating factors.

Module	Findings by the test team
AEP Keyper	1. Issue Security Officer smartcards 2. Set the HSM state to operational 3. Setup clock and network configuration 4. Issue Operator smartcards 5. Select FIPS mode 6. Set the HSM to online state 7. Create PKCS#11 token(s)
Safenet Luna	1. Set up network connections and configuration (time, NTP) 2. Initialize the HSM, create HSM admin PED key(s), configure m-of-n. 3. Edit HSM configuration/policies 4. Create partition(s) including partition owner(s) and PED keys. 5. Exchange certificates between HSM and client to set up Network Trust Link (NTL). 6. Setup client application to use the partition as a PKCS#11 token
Thales nShield	1. Create a Security World and set FIPS mode 2. Create an Administrator Card Set for the Security World 3. Create Operator Card Set(s) 4. Verify that the Security World was set up correctly. 5. Setup client application to use an OCS as PKCS#11 token
Utimaco CryptoServer	1. Set up network configuration 2. Design the permission structure 3. Replace the default admin account 4. Create new users and set privileges 5. Create PKCS#11 token(s).

C.6 Multiple security domains

Module	Findings by the test team
AEP Keyper	It is possible to have multiple PKCS#11 tokens but there is no way of dividing the HSM into separately administered partitions or similar.
Safenet Luna	The HSM can be divided into up to 20 partitions with separate administrators.
Thales nShield	Operator Card Sets represent independent security domains with no cryptographic interconnection. There is a direct equivalence between an OCS and a PKCS#11 token.
Utimaco CryptoServer	There is no way of splitting the HSM into separately administrable partitions. But each PKCS#11 token can have dedicated user(s) assigned to it, so the administration of PKCS#11 tokens can be separated.

C.7 Time synchronization – A requirement for auditing and non-repudiation is that logged messages include a time and date from a protected source.

Module	Findings by the test team
AEP Keyper	Authenticated change of time and date. No NTP support.
Safenet Luna	Authenticated change of time and date. Support for Secure NTP.
Thales nShield	Change of time and date on front panel. No NTP support.
Utimaco CryptoServer	Authenticated change of time and date. Support for NTP

4.4 Status and monitoring (E)

E.1 Which event types are logged?

Module	Findings by the test team
AEP Keyper	Logging for HSM administration, usage and application operations.
Safenet Luna	All daemons running on the HSM logs to syslog.
Thales nShield	Logging for HSM administration, usage and application operations.
Utimaco CryptoServer	Logging is configured per application running on the HSM, such as the network daemon, the NTP daemon etc.

E.2 How are logs stored and retrieved?

Module	Findings by the test team
AEP Keyper	An audit log is stored on the HSM and can be retrieved unauthenticated from any client on the same network if the HSM is in online state. Interpretation of audit log codes is provided in the user documentation.
Safenet Luna	Logs are stored in the syslog on the HSM and can be retrieved using an SCP-like command line tool.
Thales nShield	Log files are stored on board the HSM and can be viewed via the front panel and/or synchronized with the RFS.
Utimaco CryptoServer	The logs are stored in an on board file which can be downloaded to the administration client.

E.3 SNMP support

Module	Findings by the test team
AEP	No support for SNMP.
Safenet Luna	Partial or full support for most of the MIBs defined in RFCs 3410-3418.
Thales nShield	Yes, a MIB is shipped as part of the software package.
Utimaco CryptoServer	Yes, a MIB is shipped as part of the software package.

4.5 Usability (F)

F.1 Setup and ease of use

Module	Findings by the test team
AEP Keyper	Setup and initialization is impressively simple. The security model is transparent and easy to understand. Lack of remote administration features might make the Keyper unsuitable for certain operation environments.
Safenet Luna	Setup and initialization of the HSM is relatively straightforward and the security model understandable. The use of PED keys of different colours was slightly confusing. However the PED does a good job in abstracting the use of a smartcard reader while still enabling remote command authorization.
Thales nShield	Setup and initialization of the HSM for a first time user is slightly complicated since it requires understanding the Security World and Remote File System model. The actual steps are straightforward though. It is convenient to have an automated software backup of the HSM through the RFS.
Utimaco CryptoServer	It was fairly easy to get the HSM into running state. However the CLI syntax for issuing commands is complicated and takes time getting used to. For smartcard authentication it is necessary to specify card type, reader type and actual USB/serial port being used. The permission model is unnecessarily complex for most applications, which might encourage administrators to use an insecure default configuration instead.

F.2 Physical administration interface

Module	Findings by the test team
AEP Keyper	Efficient front panel administration. The form factor of the unit differs quite drastically from standard rack appliances.
Safenet Luna	No administration directly on the HSM front panel.
Thales nShield	Front panel administration with modern display and convenient navigation features.
Utimaco CryptoServer	Front panel administration with modern display and convenient navigation features.

F.3 Software administration interfaces

Module	Findings by the test team
AEP Keyper	As mentioned in B.2 most administration tasks are done directly on the unit.
Safenet Luna	Command line administration over SSH or direct serial connection. It is possible to use either a command shell or a few bundled CLI tools.
Thales nShield	Windows/Linux Java GUI and CLI. The Windows GUI needs a small work over to be fully compliant with Windows 7/Windows 2008 Server.
Utimaco CryptoServer	Windows/Linux Java GUI and CLI.

F.4 Documentation

Module	Findings by the test team
AEP Keyper	Very clear and security oriented documentation. The visual format could have been more appealing but the text is informative and to the point.
Safenet Luna	Web based documentation only. The quick start and installation guides are easy to follow. More advanced features are documented but somewhat hard to find. A better structured documentation layout with introductory text to new concepts would have been advisable.
Thales nShield	Informative user guide. Whitepapers describing the rationale behind the Security World/RFS architecture and PKCS#11 integration would have facilitated initial deployment.
Utimaco CryptoServer	The documentation is comprehensive but split into several documents and the structure if somewhat unintuitive. For instance it took time to realize that the "quick start guide" was located in one of the last chapters in the admin guide.

5 Key Findings

5.1 Summary

Generally all the modules work as expected and offer the necessary functionality one needs from a secure crypto processor. No stability issues or crashes were encountered during testing. Which HSM to choose depends on budget, the deployment scenario, performance requirements and application specific facts.

We were surprised to find an unexpected level of diversity in how the respective vendors had chosen to implement features such as role structure, authorization models and key backup. This fact made it fairly difficult to get started with each new HSM. The documentation was not always very helpful but seemed rather to be written as a reference for someone already having product specific knowledge.

We have to conclude that in order to achieve an effective HSM deployment it is probably necessary to involve vendor product experts. It is hard to find or define a best practise over such a diverse flora of appliances and usage models.

From an application perspective however the PKCS#11 interface worked exemplary on all modules. Once set up we hardly experienced any problems with the interface. The only issue worth mentioning is the fact that we needed to execute several concurrent threads (for all modules) in order to achieve a decent HSM CPU load.

5.2 Suggested future work

The test team would have appreciated a more standardized security and authorization model and nomenclature. As it is now each vendors has chosen to integrate with the PKCS#11 model in different fashions. An evolvement of the PKCS#11 standard to incorporate more complex modules than smartcards would probably be advisable.

When performing this review it would have been very helpful to have had access to best practise recommendations for setting up and configuring HSMs. Such a text could also document specific application areas and general deployment scenarios. At the moment the user is referred primarily to vendor specific whitepapers and presentations.

It would be interesting to do a more formal HSM penetration test and security analysis of a complete solution including parts not validated in the FIPS 140-2 process. One of the goals of such a test could be to derive appropriate recommendations for configuring a HSM system for certain deployment scenarios.

6 Acknowledgements

Rickard Bellgrim from the OpenDNSSEC development team has provided valuable feedback and practical PKCS#11 testing tools. Karl Castor assisted practically during the testing phase.

7 Abbreviations

7.1 General

AES	– Advanced Encryption Standard
CAPI	– (Microsoft) Cryptographic API
CLI	– Command Line Interface
CPU	– Central Processing Unit
DSA	– Digital Signature Algorithm
DNSSEC	– DNS Security Extensions
FIPS	– Federal Information Processing Standard
GUI	– User Interface
HSM	– Hardware Security Module
HA	– High Availability
JCA/JCE	– Java Cryptography Architecture/Extension
LAN	– Local Area Network
MIB	– Management Information Base
NTP	– Network Time Protocol
NVRAM	– Non-volatile Random Access Memory
PCI	– Peripheral Component Interconnect
PCMCIA	– Personal Computer Memory Card International Association
PED	– PIN Entry Device
PIN	– Personal Identification Number
PKCS	– Public Key Cryptography Standards
PKI	– Public Key Infrastructure
RSA	– Rivest Shamir Adelman
SCP	– Secure Copy
SHA	– Secure Hash Algorithm
SNMP	– Secure Network Monitoring Protocol
SSH	– Secure Shell
SSL	– Secure Sockets Layer

7.2 Vendor specific

7.2.1 AEP

AAK	– Adaptor Authorization Key
SMK	– Storage Master Key

7.2.2 Safenet

NTL	– Network Trust Link

7.2.3 Thales

ACS	– Administrator Card Set
CHIL	– Cryptographic Hardware Interface Library
OCS	– Operator Card Set
RFS	– Remote File System
SEE	– Secure Execution Engine

7.2.4 Utimaco

CXI	– Cryptographic eXtended services Interface
MBK	– Master Backup Key

8 References

[FIPS140-2] NIST, Security Requirements for Cryptographic Modules, http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf 2002

[FIPSVal] NIST, http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm

[NSASuiteB] NIST, http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml

[OpenDNSSEC] OpenDNSSEC, www.opendnssec.org

[PKCS#11] RSA Laboratories, PKCS#11 V.220 Core Specification, ftp://ftp.rsa.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20.pdf 2004

Appendix A - Performance test details

Below we list the minimum number of threads and iterations required to achieve a stable, maximum performance for each HSM.

For the single threaded tests we used 10 000 iterations for all the modules.

For the multithreaded testing the details are provided in the table below. It should be noted that the performance was relatively stable for lower values of both thread count and iterations.

Type	AEP Keyper	Safenet Luna	Thales nShield	Utimaco CryptoServer
Threads	20	50	50	10 processes*
Iterations	10 000	10 000	10 000	10 000

*As described in review point A.5 we experienced no improved performance by running multiple threads for the CryptoServer.

Appendix B - PKCS#11 mechanisms

Supported PKCS#11 mechanisms per HSM are listed below. The information was extracted through each module's PKCS#11 interface, Hexadecimal mechanism names indicates proprietary, non-standard mechanisms. The first column lists the mechanism descriptor, the optional second column supported key sizes and the third column available operations. The test team has not verified that all of the listed mechanisms are indeed implemented.

AEP Keyper

CKM_RSA_PKCS	512 - 4096	HW encrypt decrypt sign verify wrap unwrap
CKM_RSA_9796	512 - 4096	HW sign verify
CKM_RSA_X_509	512 - 4096	HW encrypt decrypt sign verify
CKM_DSA	512 - 2048	HW sign verify
CKM_DSA_SHA1	512 - 2048	HW sign verify
CKM_DES_CBC		HW encrypt decrypt
CKM_DES_ECB		HW encrypt decrypt wrap unwrap
CKM_DES_MAC		HW sign verify
CKM_DES_CBC_PAD		HW encrypt decrypt wrap unwrap
CKM_DES3_CBC		HW encrypt decrypt
CKM_DES3_ECB		HW encrypt decrypt wrap unwrap
CKM_DES3_MAC		HW sign verify
CKM_DES3_CBC_PAD		HW encrypt decrypt wrap unwrap
CKM_KEY_WRAP_SET_OAEP		HW wrap unwrap
CKM_SHA_1_HMAC		HW sign verify
CKM_SHA_1_HMAC_GENERAL		HW sign verify
CKM_AES_CBC		HW encrypt decrypt
CKM_AES_ECB		HW encrypt decrypt wrap unwrap
CKM_AES_MAC		HW sign verify
CKM_AES_CBC_PAD		HW encrypt decrypt wrap unwrap
CKM_RSA_X9_31	512 - 4096	HW encrypt decrypt sign verify
CKM_RSA_PKCS_PSS	512 - 4096	HW encrypt decrypt sign verify
CKM_SHA_1		No-HW digest
CKM_MD5		No-HW digest
CKM_SHA256		No-HW digest
CKM_SHA384		No-HW digest
CKM_SHA512		No-HW digest
CKM_SHA224		No-HW digest
CKM_RSA_PKCS_KEY_PAIR_GEN	512 - 4096	HW generate-key-pair
CKM_DSA_KEY_PAIR_GEN	512 - 2048	HW generate-key-pair
CKM_DH_PKCS_KEY_PAIR_GEN	512 - 2048	HW generate-key-pair
CKM_DH_PKCS_DERIVE		HW derive
0x80000003		HW derive
0x80000002		HW derive
CKM_DES_KEY_GEN		HW generate
CKM_DES2_KEY_GEN		HW generate
CKM_DES3_KEY_GEN		HW generate
CKM_PBE_SHA1_DES2_EDE_CBC		HW generate
CKM_PBE_SHA1_DES3_EDE_CBC		HW generate
CKM_AES_KEY_GEN		HW generate
CKM_XOR_BASE_AND_DATA		HW derive

Safenet Luna

CKM_RSA_PKCS	256 - 4096	HW encrypt decrypt sign verify wrap unwrap
CKM_RSA_X_509	256 - 4096	HW encrypt decrypt
CKM_RSA_PKCS_KEY_PAIR_GEN	256 - 4096	HW generate-key-pair
CKM_SHA1_RSA_PKCS	256 - 4096	HW sign verify
CKM_SHA224_RSA_PKCS	256 - 4096	HW sign verify
CKM_SHA256_RSA_PKCS	256 - 4096	HW sign verify
CKM_SHA384_RSA_PKCS	256 - 4096	HW sign verify
CKM_SHA512_RSA_PKCS	256 - 4096	HW sign verify
CKM_RSA_PKCS_OAEP	256 - 4096	HW encrypt decrypt wrap unwrap
CKM_RSA_X9_31_KEY_PAIR_GEN	1024 - 4096	HW generate-key-pair
CKM_SHA1_RSA_X9_31	1024 - 4096	HW sign verify
CKM_RSA_PKCS_PSS	256 - 4096	HW sign verify
CKM_SHA1_RSA_PKCS_PSS	256 - 4096	HW sign verify
CKM_SHA224_RSA_PKCS_PSS	512 - 4096	HW sign verify
CKM_SHA256_RSA_PKCS_PSS	512 - 4096	HW sign verify
CKM_SHA384_RSA_PKCS_PSS	512 - 4096	HW sign verify
CKM_SHA512_RSA_PKCS_PSS	1024 - 4096	HW sign verify
CKM_DSA_KEY_PAIR_GEN	512 - 1024	HW generate-key-pair
CKM_DSA	512 - 1024	HW sign verify
CKM_DSA_SHA1	512 - 1024	HW sign verify
CKM_DES3_MAC	128 - 192	HW sign verify
CKM_DES3_MAC_GENERAL	128 - 192	HW sign verify
CKM_DES2_KEY_GEN	128	HW generate
CKM_DES3_KEY_GEN	192	HW generate
CKM_AES_CBC	16 - 32	HW encrypt decrypt wrap unwrap
CKM_AES_CBC_PAD	16 - 32	HW encrypt decrypt wrap unwrap
CKM_AES_ECB	16 - 32	HW encrypt decrypt wrap unwrap
CKM_AES_ECB_ENCRYPT_DATA	16 - 32	HW derive
CKM_AES_CBC_ENCRYPT_DATA	16 - 32	HW derive
CKM_AES_MAC	16 - 32	HW sign verify
CKM_AES_MAC_GENERAL	16 - 32	HW sign verify
CKM_AES_KEY_GEN	16 - 32	HW generate
CKM_SHA_1		HW digest
CKM_SHA_1_HMAC	8 - 4096	HW sign verify
CKM_SHA_1_HMAC_GENERAL	8 - 4096	HW sign verify
CKM_SHA224		HW digest
CKM_SHA224_HMAC	8 - 4096	HW sign verify
CKM_SHA224_HMAC_GENERAL	8 - 4096	HW sign verify
CKM_SHA256		HW digest
CKM_SHA256_HMAC	8 - 4096	HW sign verify
CKM_SHA256_HMAC_GENERAL	8 - 4096	HW sign verify
CKM_SHA384		HW digest
CKM_SHA384_HMAC	8 - 4096	HW sign verify
CKM_SHA384_HMAC_GENERAL	8 - 4096	HW sign verify
CKM_SHA512		HW digest
CKM_SHA512_HMAC	8 - 4096	HW sign verify
CKM_SHA512_HMAC_GENERAL	8 - 4096	HW sign verify
CKM_MD5_RSA_PKCS	256 - 4096	HW sign verify
CKM_DH_PKCS_KEY_PAIR_GEN	512 - 2048	HW generate-key-pair
CKM_DH_PKCS_DERIVE	512 - 2048	HW derive
CKM_DES_CBC	40 - 64	HW encrypt decrypt wrap unwrap
CKM_DES_CBC_PAD	40 - 64	HW encrypt decrypt wrap unwrap
CKM_DES_ECB	40 - 64	HW encrypt decrypt wrap unwrap

CKM_DES_ECB_ENCRYPT_DATA	40 - 64	HW derive
CKM_DES_CBC_ENCRYPT_DATA	40 - 64	HW derive
CKM_DES_MAC	40 - 64	HW sign verify
CKM_DES_MAC_GENERAL	40 - 64	HW sign verify
CKM_DES_KEY_GEN	40 - 64	HW generate
CKM_RC2_CBC	1 - 1024	HW encrypt decrypt wrap unwrap
CKM_RC2_CBC_PAD	1 - 1024	HW encrypt decrypt wrap unwrap
CKM_RC2_ECB	1 - 1024	HW encrypt decrypt wrap unwrap
CKM_RC2_MAC	1 - 1024	HW sign verify
CKM_RC2_MAC_GENERAL	1 - 1024	HW sign verify
CKM_RC2_KEY_GEN	1 - 1024	HW generate
CKM_RC4	8 - 2048	HW encrypt decrypt
CKM_RC4_KEY_GEN	8 - 2048	HW generate
CKM_RC5_CBC		HW encrypt decrypt
CKM_RC5_CBC_PAD		HW encrypt decrypt
CKM_RC5_ECB		HW encrypt decrypt
CKM_RC5_MAC		HW sign verify
CKM_RC5_MAC_GENERAL		HW sign verify
CKM_RC5_KEY_GEN		HW generate
CKM_CAST_CBC	40 - 64	HW encrypt decrypt wrap unwrap
CKM_CAST_CBC_PAD	40 - 64	HW encrypt decrypt wrap unwrap
CKM_CAST_ECB	40 - 64	HW encrypt decrypt wrap unwrap
CKM_CAST_MAC	40 - 64	HW sign verify
CKM_CAST_MAC_GENERAL	40 - 64	HW sign verify
CKM_CAST_KEY_GEN	40 - 64	HW generate
CKM_CAST3_CBC_PAD	40 - 64	HW encrypt decrypt wrap unwrap
CKM_CAST3_CBC	40 - 64	HW encrypt decrypt wrap unwrap
CKM_CAST3_ECB	40 - 64	HW encrypt decrypt wrap unwrap
CKM_CAST3_MAC	40 - 64	HW sign verify
CKM_CAST3_MAC_GENERAL	40 - 64	HW sign verify
CKM_CAST3_KEY_GEN	40 - 64	HW generate
CKM_CAST128_CBC	40 - 128	HW encrypt decrypt wrap unwrap
CKM_CAST128_CBC_PAD	40 - 128	HW encrypt decrypt wrap unwrap
CKM_CAST128_ECB	40 - 128	HW encrypt decrypt wrap unwrap
CKM_CAST128_MAC	40 - 128	HW sign verify
CKM_CAST128_MAC_GENERAL	40 - 128	HW sign verify
CKM_CAST128_KEY_GEN	40 - 128	HW generate
CKM_MD2		HW digest
CKM_MD2_KEY_DERIVATION	8 - 128	HW derive
CKM_PBE_MD2_DES_CBC	64	HW generate
CKM_MD5		HW digest
CKM_SSL3_MD5_MAC	128	HW sign verify
CKM_MD5_HMAC	8 - 4096	HW sign verify
CKM_MD5_HMAC_GENERAL	8 - 4096	HW sign verify
CKM_MD5_KEY_DERIVATION	8 - 128	HW derive
CKM_PBE_MD5_DES_CBC	64	HW generate
CKM_PBE_MD5_CAST_CBC	64	HW generate
CKM_PBE_MD5_CAST3_CBC	64	HW generate
CKM_PBE_SHA1_CAST128_CBC	40 - 128	HW generate
CKM_SSL3_MASTER_KEY_DERIVE	384	HW derive
CKM_SSL3_KEY_AND_MAC_DERIVE	384	HW derive
CKM_SSL3_PRE_MASTER_KEY_GEN	384	HW generate
CKM_SSL3_SHA1_MAC	160	HW sign verify
CKM_CONCATENATE_BASE_AND_KEY	8 - 4096	HW derive
0x0000800D	8 - 4096	HW derive
CKM_CONCATENATE_BASE_AND_DATA	8 - 4096	HW derive

CKM_CONCATENATE_DATA_AND_BASE	8 - 4096	HW derive
CKM_XOR_BASE_AND_DATA	8 - 4096	HW derive
0x8000001B	8 - 4096	HW derive
CKM_EXTRACT_KEY_FROM_KEY	8 - 4096	HW derive
CKM_GENERIC_SECRET_KEY_GEN	8 - 4096	HW generate
CKM_SHA1_KEY_DERIVATION	8 - 160	HW derive
CKM_SHA224_KEY_DERIVATION	8 - 160	HW derive
CKM_SHA256_KEY_DERIVATION	8 - 160	HW derive
CKM_SHA384_KEY_DERIVATION	8 - 160	HW derive
CKM_SHA512_KEY_DERIVATION	8 - 160	HW derive
CKM_PBE_SHA1_RC4_128	128	HW generate
CKM_PBE_SHA1_RC4_40	40	HW generate
CKM_PBE_SHA1_DES3_EDE_CBC	192	HW generate
CKM_PBE_SHA1_DES2_EDE_CBC	128	HW generate
CKM_PBE_SHA1_RC2_128_CBC	128	HW generate
CKM_PBE_SHA1_RC2_40_CBC	40	HW generate
CKM_PKCS5_PBKD2	1 - 512	HW generate
CKM_ARIA_CBC	16 - 32	HW encrypt decrypt wrap unwrap
CKM_ARIA_CBC_PAD	16 - 32	HW encrypt decrypt wrap unwrap
CKM_ARIA_ECB	16 - 32	HW encrypt decrypt wrap unwrap
CKM_ARIA_ECB_ENCRYPT_DATA	16 - 32	HW derive
CKM_ARIA_CBC_ENCRYPT_DATA	16 - 32	HW derive
CKM_ARIA_MAC	16 - 32	HW sign verify
CKM_ARIA_MAC_GENERAL	16 - 32	HW sign verify
CKM_ARIA_KEY_GEN	16 - 32	HW generate
0x80000105	128	HW encrypt decrypt wrap unwrap
0x80000106	128	HW encrypt decrypt wrap unwrap
0x80000104	128	HW encrypt decrypt wrap unwrap
0x80000107	128	HW sign verify
0x80000103	128	HW generate
0x80000100		HW digest
0x80000101	1024 - 2048	HW generate-key-pair
0x80000102	1024 - 2048	HW sign verify
0x80000109	1024 - 2048	HW sign verify
CKM_ECDH1_DERIVE	112 - 571	HW derive
CKM_DES3_CBC	128 - 192	HW encrypt decrypt wrap unwrap
CKM_DES3_CBC_PAD	128 - 192	HW encrypt decrypt wrap unwrap
CKM_DES3_ECB	128 - 192	HW encrypt decrypt wrap unwrap
CKM_DES3_ECB_ENCRYPT_DATA	128 - 192	HW derive
CKM_DES3_CBC_ENCRYPT_DATA	128 - 192	HW derive
CKM_EC_KEY_PAIR_GEN	112 - 571	HW generate-key-pair
CKM_ECDSA	112 - 571	HW sign verify
CKM_ECDSA_SHA1	112 - 571	HW sign verify

Thales nShield

CKM_RSA_PKCS_KEY_PAIR_GEN	16 - 4096	HW generate-key-pair
CKM_RSA_X9_31_KEY_PAIR_GEN	16 - 4096	HW generate-key-pair
CKM_RSA_PKCS	16 - 4096	HW encrypt decrypt sign verify wrap unwrap
CKM_RSA_9796	16 - 4096	HW sign verify
CKM_RSA_X_509	16 - 4096	HW encrypt decrypt sign verify
CKM_RSA_PKCS_OAEP	16 - 4096	HW encrypt decrypt wrap unwrap
CKM_RSA_PKCS_PSS	16 - 4096	HW sign verify
CKM_SHA1_RSA_PKCS_PSS	16 - 4096	HW sign verify
CKM_SHA224_RSA_PKCS_PSS	16 - 4096	HW sign verify
CKM_SHA256_RSA_PKCS_PSS	16 - 4096	HW sign verify
CKM_SHA384_RSA_PKCS_PSS	16 - 4096	HW sign verify
CKM_SHA512_RSA_PKCS_PSS	16 - 4096	HW sign verify
CKM_MD5_RSA_PKCS	27 - 4096	HW sign verify
CKM_SHA1_RSA_PKCS	31 - 4096	HW sign verify
CKM_SHA256_RSA_PKCS	31 - 4096	HW sign verify
CKM_SHA384_RSA_PKCS	31 - 4096	HW sign verify
CKM_SHA512_RSA_PKCS	31 - 4096	HW sign verify
CKM_DSA_KEY_PAIR_GEN	512 - 2048	HW generate-key-pair
CKM_DSA	512 - 2048	HW sign verify
CKM_DSA_SHA1	512 - 2048	HW sign verify
CKM_DSA_PARAMETER_GEN	1024 - 2048	HW generate
0xDE43698A	1024 - 2048	HW sign verify
CKM_DH_PKCS_KEY_PAIR_GEN	16 - 4096	HW generate-key-pair
CKM_DH_PKCS_DERIVE	16 - 4096	HW derive
CKM_DES_KEY_GEN	8	HW generate
CKM_DES_ECB		HW encrypt decrypt wrap unwrap derive
CKM_DES_ECB_ENCRYPT_DATA		HW derive
CKM_DES_CBC		HW encrypt decrypt wrap unwrap
CKM_DES_CBC_PAD		HW encrypt decrypt wrap unwrap
CKM_DES_MAC		HW sign verify
CKM_DES_MAC_GENERAL		HW sign verify
CKM_DES2_KEY_GEN	16	HW generate
CKM_DES3_KEY_GEN	24	HW generate
CKM_DES3_ECB		HW encrypt decrypt wrap unwrap derive
CKM_DES3_ECB_ENCRYPT_DATA		HW derive
CKM_DES3_CBC_ENCRYPT_DATA		HW derive
CKM_DES3_CBC		HW encrypt decrypt wrap unwrap
CKM_DES3_CBC_PAD		HW encrypt decrypt wrap unwrap
CKM_DES3_MAC		HW sign verify
CKM_DES3_MAC_GENERAL		HW sign verify
CKM_AES_KEY_GEN	16 - 32	HW generate
CKM_AES_ECB		HW encrypt decrypt wrap unwrap derive
CKM_AES_CBC		HW encrypt decrypt wrap unwrap
CKM_AES_CBC_PAD		HW encrypt decrypt wrap unwrap
CKM_AES_MAC		HW sign verify
CKM_AES_MAC_GENERAL		HW sign verify
0xDE4379F9		HW sign verify
CKM_MD5		No-HW digest
0xDE436978	1 - 2000	HW generate
CKM_MD5_HMAC	1 - 2000	HW sign verify
CKM_MD5_HMAC_GENERAL	1 - 2000	HW sign verify
CKM_SHA_1		No-HW digest
CKM_SHA224		No-HW digest

CKM_SHA256		No-HW digest
CKM_SHA384		No-HW digest
CKM_SHA512		No-HW digest
CKM_RIPEMD160		No-HW digest
0xDE436975	1 - 2000	HW generate
CKM_SHA_1_HMAC	1 - 2000	HW sign verify
CKM_SHA_1_HMAC_GENERAL	1 - 2000	HW sign verify
CKM_GENERIC_SECRET_KEY_GEN	1 - 2000	HW generate
CKM_XOR_BASE_AND_DATA		HW derive
CKM_CONCATENATE_BASE_AND_KEY		HW derive
0xDE438A72		HW derive
CKM_PBE_MD5_DES_CBC		HW generate
0xDE436973		HW wrap
0xDE436974		HW generate

Utimaco CryptoServer

CKM_DES_ECB	8 - 24	HW encrypt decrypt wrap unwrap
CKM_DES3_ECB	8 - 24	HW encrypt decrypt wrap unwrap
CKM_DES_CBC	8 - 24	HW encrypt decrypt wrap unwrap
CKM_DES3_CBC	8 - 24	HW encrypt decrypt wrap unwrap
CKM_DES_CBC_PAD	8 - 24	HW encrypt decrypt wrap unwrap
CKM_DES3_CBC_PAD	8 - 24	HW encrypt decrypt wrap unwrap
CKM_AES_ECB	16 - 32	HW encrypt decrypt wrap unwrap
CKM_AES_CBC	16 - 32	HW encrypt decrypt wrap unwrap
CKM_AES_CBC_PAD	16 - 32	HW encrypt decrypt wrap unwrap
CKM_AES_CTR	16 - 32	HW encrypt decrypt
CKM_MD5		HW digest
CKM_SHA_1		HW digest
CKM_SHA224		HW digest
CKM_SHA256		HW digest
CKM_SHA384		HW digest
CKM_SHA512		HW digest
CKM_RIPEMD160		HW digest
CKM_DES_MAC	8 - 24	HW sign verify
CKM_DES_MAC_GENERAL	8 - 24	HW sign verify
CKM_DES3_MAC	8 - 24	HW sign verify
CKM_DES3_MAC_GENERAL	8 - 24	HW sign verify
0x80000135	8 - 24	HW sign verify
CKM_AES_MAC	16 - 32	HW sign verify
CKM_AES_MAC_GENERAL	16 - 32	HW sign verify
CKM_SHA_1_HMAC	1 - 1024	HW sign verify
CKM_SHA_1_HMAC_GENERAL	1 - 1024	HW sign verify
CKM_MD5_HMAC	1 - 1024	HW sign verify
CKM_MD5_HMAC_GENERAL	1 - 1024	HW sign verify
CKM_RIPEMD160_HMAC	1 - 1024	HW sign verify
CKM_RIPEMD160_HMAC_GENERAL	1 - 1024	HW sign verify
CKM_SHA256_HMAC	1 - 1024	HW sign verify
CKM_SHA256_HMAC_GENERAL	1 - 1024	HW sign verify
CKM_SHA384_HMAC	1 - 1024	HW sign verify
CKM_SHA384_HMAC_GENERAL	1 - 1024	HW sign verify
CKM_SHA512_HMAC	1 - 1024	HW sign verify
CKM_SHA512_HMAC_GENERAL	1 - 1024	HW sign verify
CKM_SHA224_HMAC	1 - 1024	HW sign verify
CKM_SHA224_HMAC_GENERAL	1 - 1024	HW sign verify
CKM_RSA_PKCS	512 - 8192	HW encrypt decrypt sign verify wrap unwrap
CKM_RSA_X_509	512 - 8192	HW encrypt decrypt sign verify
CKM_RSA_X9_31	512 - 8192	HW sign verify
CKM_RSA_PKCS_OAEP	512 - 8192	HW encrypt decrypt wrap unwrap
CKM_RSA_PKCS_PSS	512 - 8192	HW sign verify
CKM_SHA1_RSA_PKCS	512 - 8192	HW sign verify
CKM_SHA1_RSA_PKCS_PSS	512 - 8192	HW sign verify
CKM_SHA1_RSA_X9_31	512 - 8192	HW sign verify
CKM_SHA224_RSA_PKCS	512 - 8192	HW sign verify
CKM_SHA224_RSA_PKCS_PSS	512 - 8192	HW sign verify
CKM_SHA256_RSA_PKCS	512 - 8192	HW sign verify
CKM_SHA256_RSA_PKCS_PSS	512 - 8192	HW sign verify
CKM_SHA384_RSA_PKCS	512 - 8192	HW sign verify
CKM_SHA384_RSA_PKCS_PSS	512 - 8192	HW sign verify
CKM_SHA512_RSA_PKCS	512 - 8192	HW sign verify

CKM_SHA512_RSA_PKCS_PSS	512 - 8192	HW sign verify
CKM_RIPEMD160_RSA_PKCS	512 - 8192	HW sign verify
CKM_MD5_RSA_PKCS	512 - 8192	HW sign verify
CKM_ECDSA	112 - 521	HW sign verify
CKM_ECDSA_SHA1	112 - 521	HW sign verify
0x80001042	112 - 521	HW sign verify
0x80001043	112 - 521	HW sign verify
0x80001044	112 - 521	HW sign verify
0x80001045	112 - 521	HW sign verify
0x8000104A	112 - 521	HW sign verify
CKM_DSA	512 - 4096	HW sign verify
CKM_DSA_SHA1	512 - 4096	HW sign verify
0x80002042	512 - 4096	HW sign verify
0x80002043	512 - 4096	HW sign verify
0x80002044	512 - 4096	HW sign verify
0x80002045	512 - 4096	HW sign verify
0x8000204A	512 - 4096	HW sign verify
CKM_DES_KEY_GEN	8	HW generate
CKM_DES2_KEY_GEN	16	HW generate
CKM_DES3_KEY_GEN	24	HW generate
CKM_AES_KEY_GEN	16 - 32	HW generate
CKM_GENERIC_SECRET_KEY_GEN	8 - 8192	HW generate
CKM_RSA_PKCS_KEY_PAIR_GEN	512 - 8192	HW generate-key-pair
CKM_RSA_X9_31_KEY_PAIR_GEN	512 - 8192	HW generate-key-pair
CKM_EC_KEY_PAIR_GEN	112 - 521	HW generate-key-pair
CKM_DSA_KEY_PAIR_GEN	512 - 4096	HW generate-key-pair
CKM_DH_PKCS_KEY_PAIR_GEN	512 - 4096	HW generate-key-pair
CKM_X9_42_DH_KEY_PAIR_GEN	512 - 4096	HW generate-key-pair
CKM_DSA_PARAMETER_GEN	512 - 4096	HW generate
CKM_X9_42_DH_PARAMETER_GEN	512 - 4096	HW generate
CKM_DES_ECB_ENCRYPT_DATA		HW derive
CKM_DES3_ECB_ENCRYPT_DATA		HW derive
CKM_DES_CBC_ENCRYPT_DATA		HW derive
CKM_DES3_CBC_ENCRYPT_DATA		HW derive
CKM_AES_ECB_ENCRYPT_DATA		HW derive
CKM_AES_CBC_ENCRYPT_DATA		HW derive
CKM_SHA1_KEY_DERIVATION		HW derive
CKM_SHA224_KEY_DERIVATION		HW derive
CKM_SHA256_KEY_DERIVATION		HW derive
CKM_SHA384_KEY_DERIVATION		HW derive
CKM_SHA512_KEY_DERIVATION		HW derive
CKM_MD5_KEY_DERIVATION		HW derive
CKM_ECDH1_DERIVE	112 - 521	HW derive
CKM_ECDH1_COFACTOR_DERIVE	112 - 521	HW derive
CKM_DH_PKCS_DERIVE	512 - 4096	HW derive
CKM_X9_42_DH_DERIVE	512 - 4096	HW derive
CKM_CONCATENATE_BASE_AND_DATA		HW derive
CKM_CONCATENATE_DATA_AND_BASE		HW derive
CKM_CONCATENATE_BASE_AND_KEY		HW derive
CKM_EXTRACT_KEY_FROM_KEY		HW derive
CKM_XOR_BASE_AND_DATA		HW derive