OpenHSM: An Open key life cycle protocol for Public Key Infrastructure's Hardware Security Modules [*]

Jean Everson Martina[**1] and Tulio Cicero Salvaro de Souza[2]
Ricardo Felipe Custodio[2]

[1] University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge – CB3 0FD – United Kingdom

[2] Laboratório de Segurança em Computação (LabSEC)
Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88040-900 – Florianópolis – SC – Brasil

`Jean.Martina@cl.cam.ac.uk, salvaro@inf.ufsc.br, custodio@inf.ufsc.br`

Abstract. The private keys used in a PKI are its most important asset. Protect these keys from unauthorised use or disclosure is essential to secure a PKI. Relying parties need assurances that the private key used to sign their certificates is controlled and managed following pre-defined statement policy. Hardware Security Modules (HSM) offer physical and logical protection and should be considered for any PKI deployment. The software that manages keys inside an HSM should control all life cycle of a private key. Normally this kind of equipment implements a embedded key management protocol and this protocols are not available to public scrutiny due to industrial interests. Other important issue is that HSMs are targeted in their development to the Bank industry and not to PKI, making some important PKI issues, like, strict key usage control and a secure auditing trail, play a secondary role. This paper presents an open protocol to securely manage private keys inside HSMs. The protocol is described, analysed and discussed.

Keywords : Key management protocol, Hardware Security Modules

1 Introduction

Key management includes key establishment, rules and protocols for generating keys, and the subsequent handling of those keys. Securely manage cryptographic keys is one of the most important and resource consuming efforts to guarantee

[*] Work supported and founded by Rede Nacional de Pesquisa/Brazil
[**] Supported by CAPES Foundation/Brazil on grant #4226-05-4

the security on public key cryptosystems. It means we must have a rigid control on the life cycle of those keys and this is not a trivial task. Moreover, we can assume that a public cryptosystem can be considered as secure as the keys are secured. Taken this as a premise we should guarantee that a key is strictly secure during all events on its life cycle. A way to achieve this is by designing systems to securely create, manage, copy, and destroy private keys maintaining an audit record of all uses during the key life.

Hardware security modules are specific hardware designed to protect key against any kind of logical and physical tampering or extraction of cryptographic material from its environment. The HSMs are normally hardware that passed by certification procedure. The most widely known are FIPS 140-2 [1], a certification developed by USA's Department of Commerce, and Common Criteria [2], developed by a consortium having in mind the creation of protection profiles for such equipment. Normally these equipments implement their own key management protocols, which due to industrial concerns are not made publicly available for scrutiny, making us reasoning about their true correctness. Another important issue to the actual HSMs is their targeted development to the Bank industry and not to PKI, making some important PKI issues, like, strict key usage control and auditing, play a secondary role in the security context, normally making the HSM just a digital safe where we trow our keys.

Key management life cycle has been studied by many researchers [3–5]. Menezes et al. [6] discuss the public key management in a general context, including from user registration and initialization to key revocation.

However, protecting a private key in a CA context was always one of the main concerns in any PKI deployment, and is discussed by Jeff Schiller [7]. He states that protection schemes can be broken into two basic classes: schemes where no human ever has access to the raw private key material and schemes where a human may have access to the raw private key material. In the first, the private key is stored in a hardware device which itself requires a hardware token to operate. He advises that when a key is generated by this kind of devices, special attention should take in account to deploy facilities to recover the key from a failed unit.

Having an open protocol is an important matter when concerning to cryptographic algorithms and to cryptographic protocols. This was stated by Auguste Kerckhoffs[8] in the 19th century and by Claude Shanon[9] in 1948, and our main concern when designing this protocol is the lack of access to the industry owned protocols due to their intent to protect their copyrighted material. This makes us always suspicious when using a so sensitive equipment like a HSM to control keys for a Certification Authority in a PKI environment.

This work presents a cryptographic protocol to manage private keys. Our focus is an open key life cycle protocol for public key infrastructure's Hardware Security Modules which will fit on Schiller's first category. The proposed protocol was embedded in a hardware designed to be a HSM holding all physical tampering countermeasures.

The paper is structured with this introduction section, followed by section 2, where we present all the protocol basic ideas and concepts, as well as the premises we assumed during the protocol development in subsection 2.1. Later on section 3 we present our sub-protocol for initialisation and creation of the administrator group. In section 4 we present our sub-protocol for creation of the operators groups, addressing the problem of no trust between the groups inside the proposed HSM. Then in section 5 we present the sub-protocol for key generation and assignment to an operator group, which is followed by section 6, where we present the sub-protocol responsible to apply the concept of key policies and that will allow the operators group to unwrap a key for usage. Finally in section 7 we address some implementation issues that arose in our work and detail our prototyping environment. In section 8 we present our conclusions and propose future work in the theme.

2 Protocols

To address the problems on key life cycle management, we need to answer some questions on the key during all its life cycle. First, it is important to know who is authorised to create a key. This means that only authorised users can create keys and then delegate the use of the key to other user. In addition, we should guarantee that the key is unique and was generated on a secure and controlled environment, i.e., no one knows or has generated the same key pair before. The system administrator can answer the former questions by using a certified system. The singleness of the key can be achieved by using a true random number generator and the key has always to be stored in the memory of such certified system while not ciphered.

The precise time when the key is generated, used, or destroyed must be logged. The way the key is released to use and when and how many times it was used must be also subject of control and tracing.

To each key can be attributed a policy. The key, for instance, can be released for n uses or for a period of time for an application. Other aspect to consider is the control of how many operational copies there are for a key. As many operational keys exists, much more difficult to manage the life cycle of each copy of the key. Due to the additional difficulties introduced by a high number of operational keys, is advisable to keep as low is possible these number. In some practical situations, like a signature system, is common to keep only one key.

We will be presenting in this paper our proposal to address the problem of private key management in public key infrastructures, specifically in the domain of Hardware Security Modules. Our key ideas in the protocol development were to work under a shared responsibility scheme, were all operations must be done by groups instead of a single person, and they will have one symmetric key Ks which will be used to encrypt the asymmetrical private key Kr material assigned to them. This symmetric key will be shared between the group using a share secret algorithm, like the one proposed by Shamir [10], allowing to reconstruct the secret with a minimum number of parts specified at group creation. This is an important operation in the OpenHSM protocol, on all its uses, because by splitting the ownership of a key through a group, we increase the number of people necessary to corrupt and to exploit to gain access to the key being shared, increasing in this way the whole security of the system.

Another key feature we present in our protocol is the existence of an internal public key infrastructure, which will have a self-signed certificate issued internally by the HSM that will constitute our trust point. From this certificate, we construct a single certification authority to issue certificates to all people involved in the operations of the proposed protocol. By controlling the access to the private key of this internal CA we can limit administrative operations in our protocol, chaining the administrative task to a successful secret share reconstruction and a subsequent decryption of the private key tied with this certificate.

This paper just take in count the four initial processes of the key management scheme, that are the initialisation and creation of administrator group, the creation of operators group, the application's asymmetric key generation and application's asymmetric key usage. This is done this way due to limits on space in the paper. We also give clues on other important facts to a complete key management scheme, like the necessity of modifying the current groups, change the ownership of an application key, do secure backups to the whole system and enable an audit trail. Also because of space limitations we summarise all the descriptions for principals, message parts/objects and operations in Appendix A.

2.1 Premises

To design the proposed protocol we have established some premises as follow:

– each administrator ADM_I and operator $OPER_I$ hold securely its private keys, respectively Kr_{ADM_I} and Kr_{OPER_I};
– random number generator works perfectly and true randomly as an internal part of the principals generating cryptographic keys;
– NXD is a data storage that should be considered as any other, but its data is flushed on a pre-established basis;
– data storages, like ADS, ODS, CRL, CTL, NXD and KDS, store data as it was sent to them, no additional cryptography is used;
– the secret sharing scheme works perfectly;
– all data is securely deleted by a part that holds it when it knows that it will not be used anymore in the current run of the protocol, and no data can be kept to other runs of the protocol, except the data sent to those principals acting as storage facilities;
– all certificates used in the protocols are checked against their $CRLs$ and their certificate path must be constructed with a certificate contained in CTL as point of trust.

The premises above have the only purpose of delimitating and normally address some implementation related problem that we must assume as solved when considering the protocol.

3 Initialisation and Creation of Administrator Group

We start this process creating a system-wide administrator group by informing two values to feed the shared secret scheme, N and M, where $1 \leq N \leq M$ and they will denote the minimum number of principals in the group to activate it, and the size of the system-wide administrator group. Each individual administrator must know the key pair and the personal information that will compose its certificate that will be issued by the internal PKI in the HSM. Finally, to initiate the protocol run, the HSM can generate in advance its key pair Kr_{HSM}, Ku_{HSM} and a symmetric key Ks_{ADM}, all this initial knowledge is described in Table 1.

Table 1. Principals Initial Knowledge

Principal	**Initial Knowledge**
ADM_I	$N, M, Kr_{ADM_I}, Ku_{ADM_I}, ADM_{ID_I}$ [4]
HSM	$Kr_{HSM}, Ku_{HSM}, Ks_{HSM}$ [4]

3.1 Messages Exchange

The sub-protocol we propose to handle the initialisations, create the basic environment to the key life cycle management, and create the administrator group is as follows:

1. $ADM \longrightarrow HSM : N, M$
2. $HSM \longrightarrow CTL : \{|Ku_{HSM}|\}_{Kr_{HSM}}$
3. $ADM_I \longrightarrow HSM : Ku_{ADM_I}, ADM_{ID_I}$
4. $HSM \longrightarrow ADM_I : \{|Ku_{ADM_I}|\}_{Kr_{HSM}}, \{|Ku_{HSM}|\}_{Kr_{HSM}}$
5. $HSM \longrightarrow HSM : Ks_{ADM_1}||...||Ks_{ADM_M}$
6. $HSM \longrightarrow ADS : \{Ks_{ADM_1}\}_{Ku_{ADM_1}} ... \{Ks_{ADM_M}\}_{Ku_{ADM_M}},$
   $\{|Ku_{ADM_1}|\}_{Kr_{HSM}} ... \{|Ku_{ADM_M}|\}_{Kr_{HSM}},$
   $\{Kr_{HSM}\}_{Ks_{ADM}}$

In the step 1, an administrator from the proposed set informs the HSM his willing to initialise the process by sending N and M, that are respectively, the minimum amount of administrators to reconstruct the shared secret used to protect their session key and the size of the administrator group. In the transition between step 1 and step 2, the HSM generates a key pair, named Kr_{HSM} and Ku_{HSM}, and will generate a self-signed digital certificate in X509v3 format[11], that will mark our point of trust. To establish the trust in this certificate, in the step 2, the HSM will store this certificate in the CTL, that will be checked to

[4] Not necessarily an initial knowledge. This knowledge can be generated during the protocol run.

guarantee the validity of every principals certificate present in the subsequent protocols runs.

After this initial trust establishment, every ADM_I from the administrators groups will interact with the HSM, by generating its own key pair Kr_{ADM_I}, Ku_{ADM_I} and sending Ku_{ADM_I} and ADM_{ID_I} to the HSM as is shown in step 3, where ADM_{ID_I} is the identification needed to issue the ADM_I user certificate. By receiving Ku_{ADM_I} and ADM_{ID_I} the HSM, in step 4 will issue a certificate $\{|Ku_{ADM_I}|\}_{Kr_{HSM}}$ with the information provided, and will send this certificate, together with its own self-signed certificate $\{|Ku_{HSM}|\}_{Kr_{HSM}}$ to the administrator ADM_I that will store them properly.

After interacting with all M administrators informed in step 1, the HSM will run step 5, where it will generate randomly a session key Ks_{ADM}, and will split it using the secret sharing scheme explained in the previous section 2. After having Ks_{ADM} shared, the HSM will encrypt every $\{Ks_{ADM_I}\}$ with the public key Ku_{ADM_I} extracted from the ADM_I's certificate just generated, and will store all M encrypted parts together with $\{Kr_{HSM}\}_{Ks_{ADM}}$ in the Administrators Data Storage (ADS), as show in step 6.

3.2 Key objectives of the Sub-Protocol

As this sub-protocol is meant to initialise the HSM, establish trust points, and to create the administrators, we shall accomplish the following security objectives:

- never disclose Kr_{HSM};
- never disclose Ks_{ADM};
- never disclose Ks_{ADM_I};
- never disclose Kr_{ADM_I};

The requirement of non disclosure of Kr_{HSM} exist because this private key is used to establish all trust inside the HSM by signing all the certificates belonging to administrators, and self-signing the HSM certificate. Ks_{ADM} should never be disclosed because it is used to protect Kr_{HSM} on its storage form. Any Ks_{ADM_I} should never be disclosed because by having N or more parts, independently of order, can lead to the reconstruction of Ks_{ADM}, what will make Kr_{HSM} accessible. According to what was stated on previous sections, Kr_{ADM_I} should never be disclosed during the protocol run, because this can compromise Ks_{ADM_I} and subsequently all the rest of the security objectives of the sub-protocol.

4 Creation of Operators Group

This operation will create the operators groups, which will be responsible to use and retain the guard of the HSM managed keys. This process is started by informing the K and L values, where $1 < K < L$, by the administrator group, respectively the minimum amount of operators needed to reconstruct the shared secret, and the size of the operators group. The purpose of these secret sharing operations is the same as explained in the earlier section 2.

Normally each operators group present inside our HSM implementation will represent a Certification Authority private key being protected inside our cryptographical perimeter. This feature was introduced to let a single HSM being used independently inside the same PKI environment to represent CA in the same or different levels in the hierarchy depending just on the policy established by the PKI management team.

The creation of an operators group slightly differs from administrators group creation, because it will not have a private key directly assigned to it (in this first moment) and because of the existence of Ks_{OPER*}, that is responsible for trace when the group is valid for administrative operations.

This is necessary because we do not want to establish trust between the administrator group and any of the operators group. When Ks_{OPER*} is deleted for some operator group, no administrative task can be done to this group until the recovery of Ks_{OPER*}. This scheme is possible by the use of XOR properties, and will be also important in the future to define traceability of operational copies in the case of backups of the HSM contents.

To initialise a run of this sub-protocol, we should have some initial knowledge by each player, and this is described in Table 2. Each ADM_I should know its key pair Kr_{ADM_I}, Ku_{ADM_I}, as well as the HSM certificate Kr_{ADM_I}, Ku_{ADM_I}. The administrator group should know in advance the values of $OPER_{ID}, K$ and L, that will be the group identification, the threshold of the secret sharing reconstruction and the size of the operators group being created respectively. The HSM should be able to generate during the run the values of Ks_{OPER}, Ks_{OPER*} and at least N nonce N_I to securely authenticate the administrator group.

Table 2. Principals Initial Knowledge

Principal	**Initial Knowledge**
ADM_I	$OPER_{ID}, K, L, Kr_{ADM_I}, Ku_{ADM_I}, \{\|Ku_{HSM}\|\}_{Kr_{HSM}}$
HSM	$Ks_{OPER}, Ks_{OPER*}, N_I$[4]
$OPER_I$	$Kr_{OPER_I}, Ku_{OPER_I}, OPER_{ID_I}$[4]
CTL	$\{\|Ku_{HSM}\|\}_{Kr_{HSM}}$
ADS	$\{Ks_{ADM_1}\}_{Ku_{ADM_1}} ... \{Ks_{ADM_M}\}_{Ku_{ADM_M}}, \{Kr_{HSM}\}_{Ks_{ADM}}$, $\{\|Ku_{ADM_1}\|\}_{Kr_{HSM}} ... \{\|Ku_{ADM_M}\|\}_{Kr_{HSM}}$

Each operator $OPER_I$ should know in advance its own key pair Kr_{OPER_I}, Ku_{OPER_I} and its personal data $OPER_{ID_I}$ that will be used to issue its certificates. The principals CTL and ADS should be able to provide the data necessary to correctly authenticate the administrators group

4.1 Messages Exchange

The sub-protocol we propose to handle the creation of operators groups is the following:

1. $ADM \longrightarrow HSM : K, L, OPER_{ID}$
2. $ADS \longrightarrow HSM : \{Ks_{ADM_1}\}_{Ku_{ADM_1}} ... \{Ks_{ADM_M}\}_{Ku_{ADM_M}},$
   $\{Kr_{HSM}\}_{Ks_{ADM}}, \{|Ku_{ADM_1}|\}_{Kr_{HSM}} ... \{|Ku_{ADM_M}|\}_{Kr_{HSM}}$
3. $HSM \longrightarrow ADM_I : \{\{Ks_{ADM_I}\}_{Ku_{ADM_I}}, N_I\}_{Ku_{ADM_I}}$
4. $ADM_I \longrightarrow HSM : \{Ks_{ADM_I}\}_{N_I}$
5. $HSM \longrightarrow HSM : Ks_{ADM_1}||...||Ks_{ADM_N}$
6. $HSM \longrightarrow NXD : Ks_{OPER*}$
7. $CTL \longrightarrow HSM : \{|Ku_{HSM}|\}_{Kr_{HSM}}$
8. $HSM \longrightarrow ODS : \{Ks_{OPER} \oplus Ks_{OPER*}\}_{Ku_{HSM}}$
9. $HSM \longrightarrow HSM : Ks_{OPER_1}||...||Ks_{OPER_L}$
10. $OPER_I \longrightarrow HSM : Ku_{OPER_I}, OPER_{ID_I}$
11. $HSM \longrightarrow OPER_I : \{|Ku_{OPER_I}|\}_{Kr_{HSM}}, \{|Ku_{HSM}|\}_{Kr_{HSM}}$
12. $HSM \longrightarrow ODS : \{Ks_{OPER_1}\}_{Ku_{OPER_1}} ... \{Ks_{OPER_L}\}_{Ku_{OPER_L}},$
    $\{|Ku_{OPER_1}|\}_{Kr_{HSM}} ... \{|Ku_{OPER_L}|\}_{Kr_{HSM}}$

This process is started with the administrator group informing the HSM an operator group identification $OPER_{ID}$ and the values of K and L in the step 1. K and L values are respectively, the minimum amount of operators to reconstruct the shared secret used to protect their keys and the size of the operators group, and $OPER_{ID}$ is a unique identification for the operators group being created. As we consider this is an administrative operation, we must have administrative authorisation. To start this process, in step 2 the ADS sends to the HSM the values $\{Ks_{ADM_1}\}_{Ku_{ADM_1}} ... \{Ks_{ADM_M}\}_{Ku_{ADM_M}}$ and $\{Kr_{HSM}\}_{Ks_{ADM}}$, that is all Ks_{ADM_I} ciphered with the Ku_{ADM_I}, plus Kr_{HSM} ciphered with Ks_{ADM}. Following the step 3, the HSM will send to each ADM_I in the administrator group, its ciphered part $\{Ks_{ADM_I}\}_{Ku_{ADM_I}}$ to deciphering, plus $\{N_I\}_{Ku_{ADM_I}}$, that is a freshly generated nonce ciphered with the ADM_I public key extracted from the certificates already sent from ADS. This is done due to two reasons.

First we want to avoid replay attacks in the messages sent by the ADM back to the HSM. Second, we need a shared value to cipher Ks_{ADM_I} sent in step 4 back from ADM to HSM, because this communication normally flows outside the cryptographic barrier of the HSM, and the collection of N part of Ks_{ADM_I} can lead to the reconstruction of Ks_{ADM}. Note that we double cipher the value of $\{Ks_{ADM_I}\}_{Ku_{ADM_I}}$ to avoid the capabilities of a Dolev-Yao attacker [12] in reconstructing messages to replay. After doing this process with at least N administrators, the HSM is able to reconstruct Ks_{ADM} as shown in step 5 and consequently decipher Kr_{HSM}, accomplishing the administrators authentication and enabling the protocol to continue.

Following the protocol, the HSM will generate two symmetric encryption keys, one Ks_{OPER} that will be used to cipher every key belonging to the group being created, and Ks_{OPER*}, that is a key to enable administrative operations on the operators group and will be the base for the separation of trust between the groups. The key Ks_{OPER*}, is stored in the NXD as shown in step 6 to enable administrative operations on the group for a while. In step 7, the CTL sends to the HSM the value $\{|Ku_{HSM}|\}_{Kr_{HSM}}$ that is the self-signed certificate of the HSM. This is done because we will need to cipher the result of the XOR operation between Ks_{OPER} and Ks_{OPER*} with the public key that is on this certificate. This will tie any subsequent operation on the groups with deciphering Kr_{HSM}, that means an administrative authentication. The value of $\{Ks_{OPER} \oplus Ks_{OPER*}\}_{Ku_{HSM}}$ is stored in the Operators Data Storage ODS in step 8.

After this binding to administrative task when dealing with the operator group, we need to share Ks_{OPER} to the L operators belonging to this operator group. This task is very similar with what we have done when sharing Ks_{ADM} to the ADM group in section 3. In step 9 the HSM splits Ks_{OPER} in L parts, then in step 10, each operator will inform to the HSM is own public key Ku_{OPER_I} and $OPER_{ID_I}$ that is all the identification needed to issue the operators X509v3 certificate $\{|Ku_{OPER_I}|\}_{Kr_{HSM}}$. In step 11, the HSM sends the operators certificate, plus the HSM certificate to each operator $OPER_I$ with the message $\{|Ku_{OPER_I}|\}_{Kr_{HSM}}, \{|Ku_{HSM}|\}_{Kr_{HSM}}$.

Finally, the HSM will use the public keys present in each certificate issued to the L administrator and will cipher all the L values Ks_{OPER_I} parts with it and send them to ODS together will all operators certificates $\{|Ku_{OPER_1}|\}_{Kr_{HSM}} \cdots \{|Ku_{OPER_L}|\}_{Kr_{HSM}}$, accordingly with step 12.

4.2 Key objectives of the Sub-Protocol

As this sub-protocol is meant to create the operators group, that are the groups responsible to use the keys managed by the HSM, and taking in consideration that there is no complete trust between any groups inside the HSM, we should accomplish the following security objectives:

- never disclose Ks_{OPER};
- never disclose Ks_{ADM};
- never disclose Kr_{HSM};
- never disclose Ks_{ADM_I};
- never disclose Kr_{ADM_I};
- never disclose Kr_{OPER_I};
- never disclose Ks_{OPER_I};
- never disclose $\{Ks_{OPER} \oplus Ks_{OPER*}\}$;

As we have access to Kr_{HSM}, we have the same chain of objectives as stated in section 3, so we must protect Ks_{ADM}, Ks_{ADM_I} and Kr_{ADM_I} from disclosure. The additional security objectives we must consider now are the non disclosure Ks_{OPER} that is the symmetric key that will cipher all the keys belonging to the group being created The non disclosure on Ks_{OPER_I}, because with a K amount of them we can reconstruct Ks_{OPER}, and we shall not disclose any operator private key Kr_{OPER_I}. Finally we also must protect $\{Ks_{OPER} \oplus Ks_{OPER*}\}$, because by knowing that this is a XOR compound value, an attacker can access Ks_{OPER*} from NXD and use it with $\{Ks_{OPER} \oplus Ks_{OPER*}\}$ to obtain Ks_{OPER}, that is an already specified security objective.

5 Application's Asymmetric Key Generation

This sub-protocol explain how to create the HSM managed keys. We consider this an administrative process, however, there should exist the operators group to which this key will be delegate and this administrative operation must be authorised by the operator group with the explicit existence of Ks_{OPER*} in the NXD.

The administrators are able to recover the operators group secret using Ks_{OPER*} key from NXD and $\{Ks_{OPER} \oplus Ks_{OPER*}\}_{Ku_{HSM}}$ from ODS. When the administrators are authenticated, they recover Ks_{ADM} key. Using it, they are able to load $\{Kr_{HSM}\}_{Ks_{ADM}}$ from ADS and after they can decrypt the XOR operation result. Now, using the reversible propriety of XOR operation, the administrator group can recover the operators key applying $\{Ks_{OPER} \oplus Ks_{OPER*}\} \oplus Ks_{OPER*}$. Becoming the operators key Ks_{OPER} known.

Other particularly in this sub-protocol is the KEY_PARAM. It will specify the key properties, like algorithm and size. For example, it could be a 1024 bits RSA key pair.

The table 3 show a summary about the necessary things to execute this sub-protocol.

Table 3. Principals Initial Knowledge

Principal	**Initial Knowledge**
ADM_I	$OPER_{ID}, KEY\_ID, KEY\_PARAM, Kr_{ADM_I}, Ku_{ADM_I},$ $\{\lvert Ku_{HSM}\rvert\}_{Kr_{HSM}}$
HSM	Kr_{APP}, KU_{APP}, N_I[4]
NXD	Ks_{OPER*}
ADS	$\{Ks_{ADM_1}\}_{Ku_{ADM_1}} ... \{Ks_{ADM_M}\}_{Ku_{ADM_M}}, \{Kr_{HSM}\}_{Ks_{ADM}}$
ODS	$\{Ks_{OPER} \oplus Ks_{OPER*}\}_{Ku_{HSM}}$

The administrator group must know in advance $OPER_{ID}$, KEY_ID and KEY_PARAM, that are the operator group identification to which the keys will be delegated, and identification for the keys being generated, and the parameters for the key generation process respectively. Each ADM_I Must know its own key pair, and the HSM must be able to generate the key pair Kr_{APP}, KU_{APP} and at least N nonce N_I to securely authenticate the administrators.

The NXD must have stored Ks_{OPER*} for the group to which the keys will be delegated, showing with this that the operator group was aware of this administrative operation. ADS must have all authentication data for the administrator group stored in it, and ODS must have all authentication data for the operator group informed in $OPER_{ID}$.

5.1 Messages Exchange

The sub-protocol which creates a new HSM managed key is described below:

1. $ADM \longrightarrow HSM : KEY\_NAME, KEY\_PARAM, OPER_{ID}$
2. $ADS \longrightarrow HSM : \{Ks_{ADM_1}\}_{Ku_{ADM_1}} ... \{Ks_{ADM_M}\}_{Ku_{ADM_M}}, \{Kr_{HSM}\}_{Ks_{ADM}}$
3. $HSM \longrightarrow ADM_I : \{\{Ks_{ADM_I}\}_{Ku_{ADM_I}}, N_I\}_{Ku_{ADM_I}}$
4. $ADM_I \longrightarrow HSM : \{Ks_{ADM_I}\}_{N_I}$
5. $HSM \longrightarrow HSM : Ks_{ADM_1} || ... || Ks_{ADM_N}$
6. $ODS \longrightarrow HSM : \{Ks_{OPER} \oplus Ks_{OPER*}\}_{Ku_{HSM}}$
7. $NXD \longrightarrow HSM : Ks_{OPER*}$
8. $HSM \longrightarrow HSM : Ks_{OPER}$[5]
9. $HSM \longrightarrow OUT : Ku_{APP}$
10. $HSM \longrightarrow KDS : \{Kr_{APP}\}_{Ks_{OPER}}, Ku_{APP}, KEY\_NAME, OPER_{ID}$

This process is started with the administrator group informing the HSM the identifier of the new key KEY_NAME, the key generation parameters KEY_PARAM and the operator group identifier $OPER_{ID}$ (this group must be created before), as shown in the step 1. We set this as an administrative operation, so the administrator group must be validated. The validation is made in the steps 2, 3 and 4, where the ADS load to the HSM the data necessary to the authentication process. Thus, the HSM sends to each ADM_I its ciphered part $\{Ks_{ADM_I}\}_{Ku_{ADM_I}}$, plus a ciphered nonce $\{N_I\}_{Ku_{ADM_I}}$, and each ADM_I send back to the HSM your part ciphered with the nonce the HSM send in the previous step. Finally the administrator secret can be recovered in the 5 when it has more than N deciphered parts of Ks_{ADM}. This validation has been shown in the previous section 4.

Following the protocol, in step 6, the ODS will send to the HSM the result of the XOR operation done when creating the operators group, ciphered with Ku_{HSM}. In the next step, 7, the NXD will send the HSM the authorisation value Ks_{OPER*}, that will be used by the HSM to XOR with $\{Ks_{OPER} \oplus Ks_{OPER*}\}$ and obtain Ks_{OPER}, as shown in step 8.

Between the steps 8 and 9, the application key pair subject to the HSM protection is generated, then the public key is exported in the step 9. In step 10, all necessary information regarding the key is stored into the Key Data Storage KDS including the operators group identifier $OPER_{ID}$, key name KEY_NAME, public key and encrypted private key $\{Kr_{APP}\}_{Ks_{OPER}}, Ku_{APP}$. The operators group will use this information to allow the recovery of the key.

5.2 Key objectives of the Protocol

The main objectives in this protocol are to securely generate a key pair and delegate its usage and management to an operator group, so the main security objectives are:

[5] This is recovered by applying the XOR properties in the $\{Ks_{OPER} \oplus Ks_{OPER*}\}$

- never disclose Ks_{OPER};
- never disclose Ks_{ADM};
- never disclose Kr_{HSM};
- never disclose Kr_{APP};
- never disclose Kr_{ADM_I};
- never disclose Ks_{ADM_I};
- never disclose $\{Ks_{OPER} \oplus Ks_{OPER*}\}$;

As we consider this an administrative operation, we must have administrative authorisation to do so. By doing this way we have the same security requirements as other previous sub-protocols, that are the non disclosure of Kr_{HSM} our trust root, Ks_{ADM} its ciphering key, and Kr_{ADM_I} and Ks_{ADM_I} that are the secrets that protect Ks_{ADM} in its shared form.

Additionally, as we deal with the operator group, by doing and administrative operation in its name, we must have access to $\{Ks_{OPER} \oplus Ks_{OPER*}\}$, that when XORed with Ks_{OPER*} gives us Ks_{OPER}. They are security objectives, because they will protect Kr_{APP}, our main security goal, and mean of existence. Ks_{OPER} is important because it will cipher Kr_{APP}, and $\{Ks_{OPER} \oplus Ks_{OPER*}\}$ is important because with it we can derive easily Ks_{OPER}.

6 Application's Asymmetric Key Usage

This sub-protocol will cover the managed keys usage. This usage does not represent specifically to sign or encrypt something with the key. It will load the key and apply the specified policy on its. The operations of signing or ciphering something with the private key subject to protection is no in the scope of this protocol, and should be treated as HSM API.

The policy, represented by KEY_POL, will specify restrictions on loaded keys. The operators group can set a maximum number of operations using the key for signatures or encryptions and/or set a range of time for the key remain loaded. For example, the key can be loaded for three uses and five minutes. The first policy reach will unload the key automatically and this sub-protocol must be executed again to load the key for a next usage.

In the table 4, we can see the items which must be initial knowledge. Basically, the system must be initialised, the administrators, at least one operators group must have been created and one key must also have been created.

Table 4. Principals Initial Knowledge

Principal	**Initial Knowledge**
$OPER_I$	$KEY\_ID, KEY\_POL, Kr_{OPER_I}, Ku_{OPER_I}, \{\|Ku_{HSM}\|\}_{Kr_{HSM}}$
ODS	$\{Ks_{OPER_1}\}_{Ku_{OPER_1}} ... \{Ks_{OPER_L}\}_{Ku_{OPER_L}},$ $\{\|Ku_{OPER_1}\|\}_{Kr_{HSM}} ... \{\|Ku_{OPER_L}\|\}_{Kr_{HSM}}$
KDS	$\{Kr_{APP}\}_{Ks_{OPER}}, Ku_{APP}, OPER_{ID}$
HSM	N_I

6.1 Messages Exchange

1. $OPER \longrightarrow HSM : KEY\_ID, KEY\_POL$
2. $KDS \longrightarrow HSM : \{Kr_{APP}\}_{Ks_{OPER}}, Ku_{APP}, OPER_{ID}$
3. $ODS \longrightarrow HSM : \{Ks_{OPER_1}\}_{Ku_{OPER_1}} ... \{Ks_{OPER_L}\}_{Ku_{OPER_L}},$
   $\{|Ku_{OPER_1}|\}_{Kr_{HSM}} ... \{|Ku_{OPER_L}|\}_{Kr_{HSM}}$
4. $HSM \longrightarrow OPER_I : \{\{Ks_{OPER_I}\}_{Ku_{OPER_I}}, N_I\}_{Ku_{OPER_I}}$
5. $OPER_I \longrightarrow HSM : \{Ks_{OPER_I}\}_{N_I}$
6. $HSM \longrightarrow HSM : Ks_{OPER_1}||...||Ks_{OPER_K}$
7. $HSM \longrightarrow HSM : Kr_{APP}, KEY\_POL$

The process starts, in the step 1, when the operators group send the command to load the key including the identification KEY_ID and a policy KEY_POL for it. This is not an administrative operation, but it is necessary to validate the operators group, as they are the effective owners of the key subject to the HSM protection. The operator group to which the key belong is known because it is loaded from KDS in step 2,together with they ciphered private key $\{Kr_{APP}\}_{Ks_{OPER}}$ and its public part Ku_{APP}. The validation of the operators group will follow the same mechanism followed by the administrator group authentication in other sub-protocols.

First, in step 3 ODS will send all data necessary to authenticate the operator group, that is all ciphered parts $\{Ks_{OPER_1}\}_{Ku_{OPER_1}} ... \{Ks_{OPER_L}\}_{Ku_{OPER_L}}$, and the operators certificates $\{|Ku_{OPER_1}|\}_{Kr_{HSM}} ... \{|Ku_{OPER_L}|\}_{Kr_{HSM}}$. In step 4, the HSM will send each operator $OPER_I$ it ciphered part $\{Ks_{OPER_I}\}_{Ku_{OPER_I}}$, plus a freshly generated nonce ciphered with its public key $\{N_I\}_{Ku_{OPER_I}}$. Following, each operator $OPER_I$ will send to the HSM its part of the shared secret ciphered with the nonce $\{Ks_{OPER_I}\}_{N_I}$, as described in step 5.

After collecting K parts deciphered by the administrator, the HSM will be allowed to recover Ks_{OPER} as shown in step 6, by the reconstruction of the shared secret. Finally, the policy KEY_POL is applied in the step 7 and the key is loaded, being ready for its usage.

6.2 Key objectives of the Protocol

The main objectives in this protocol were to securely load a key pair by an operator group following a policy, so the main security objectives are:

- never disclose Ks_{OPER};
- never disclose Ks_{OPER_I};
- never disclose Kr_{APP};
- never disclose Kr_{OPER_I};

As this operation must just have authorisation from operators group, the authorisation must never have disclosure of Ks_{OPER}, the operators group secret and the key that is used to encrypt the managed keys. Kr_{OPER_I} and Ks_{OPER_I} that are the secrets that protect Ks_{OPER} in its shared form.

Additionally, as the main objective of HSM, we must never have disclosure of Kr_{APP}. It must be used just under authorisation and respecting the policy.

7 Implementation Issues

Although we create and manage keys, we do not plan any protocol for key destruction or deletion, because this could be simply accomplished by removing the ciphered parts from KDS, but this is not always that simple and sometimes extremely difficult to achieve. Normally this is covered by the PKI policies, what tend to the destruction of all private key material by physical means, like the incineration of the HSM itself and everything that could have had contact with these key material.

Other thing that is not covered in the protocol, is the operators group destruction, but this is easily achievable just by destroying X operators privates keys, where $X > N$. This will also render all keys that belonged to the operators groups set being destroyed unusable, as long the NXD part of the group is not available for administrative matters.

7.1 Prototype

The embedded application was developed in C language and involved many technologies, including smart card support, cryptography, data storage, secret sharing and X509v3 certificates. To solve these technology gaps we used known open/free libraries, such: OpenSSL - for cryptographic operations and X509 certificate support, SQLite - a lite database with focus in embedded systems, OpenSC - smart card support, SharedSecret - for sharing secrets and PCSC-LITE - supporting smart card readers and tokens, everything integrated with a FreeBSD system running with embedded customisations we developed.

The hardware was projected to be tamper proof system using a Security Unit(S.U.) to manage all sensors and protection systems, including a Random Number Generation, a separate Clock and a true eraser circuit. The key manager software was integrated using a library to manage the configuration of the S.U. and used a pipe system to receive a problem detection message from the S.U.

8 Conclusions

This work presented a cryptographic protocol to manage private keys in a Certification Authority context, i.e., an application that can be embedded in a Hardware Security Module. It is known the security of a PKI is related to how the keys are generated, used and destroyed. Thus, our protocol was conceived to have a strong control of the keys, a requirement in PKI solutions. The protocol was itself built on an internal PKI, i.e., we have designed a PKI to manage private keys of external Certification Authorities.

To manage the HSM, it was created groups. The administrator group is responsible to create new application keys and the operator groups. The latter group is bound to the private keys of the applications, like a certification authority.

The protocol was coded and embedded in a prototype HSM. The implementation has shown that it is comfortable and secure to manage private keys to Certification Authorities. It also showed that we can use this strict key controls to debug CA software. As future work it is intended to extend the protocol presented by including a backup feature and auditing system. We also propose to do formal analysis on the protocol.

References

1. FIPS: Security requirements for cryptographic modules, FIPS PUB 140-2 (2002)
2. Killmann, W., Leitold, H., Posch, R., Sallé, P.: Protection profile - secure signature-creation device type 3. http://www.commoncriteriaportal.org /public/files/ppfiles/pp0006b.pdf (July 2001)
3. Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: Recommendation for key management part 1: General. Technical Report 800-57, NIST (May 2006) NIST Special Publication.
4. Neumann, P.G.: Crypto key management. Commun. ACM **40**(8) (1997) 136
5. Daemen, J.: Management of secret keys: Dynamic key handling. In Preneel, B., Rijmen, V., eds.: State of the Art in Applied Cryptography. Volume 1528 of Lecture Notes in Computer Science., Springer (1997) 264–276
6. Menezes, A.J., Vanstone, S.A., Oorschot, P.C.V.: Handbook of Applied Cryptography. CRC Press, Inc., Boca Raton, FL, USA (1996)
7. Schiller, J.: Protecting a private key in a ca context (2000) A useful discussion of the issues and patterns.
8. Kerckhoffs, A.: La cryptographie militaire. Journal des sciences militaires **IX** (1883) 5–38
9. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal **27** (1948) 379423
10. Shamir, A.: How to share a secret. Commun. ACM **22**(11) (1979) 612–613
11. X.509, I.T.R.: Information technology - open systems interconnection - the directory: Authentication framework. Technical report, ITU-T (1997)
12. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Transactions on Information Theory **29**(2) (1983) 198–208

Appendix A: Conventions

The protocols are subject to conventions in Table 5, Table 6 and Table 7.

Table 5. Description of all Operations Used in the Protocols

Operation	Description
$\{\}$	Data inside brackets is ciphered by subscript key outside brackets.
$\{\|\|\}$	Data inside piped brackets is signed by subscript key outside brackets.
$\|\|$	Data is concatenated or dissociated using a secret sharing scheme.
$\{\oplus\}$	Data inside brackets is logically XOR-ed and the result becomes 1 single object.

Table 6. Description of all Principals of the Protocols

Principal	**Description**
ADM_I	An administrator of the HSM
HSM	The HSM Crypto-Processor
CTL	Certificate Trust List
ADS	Administrator Data Storage
$OPER_I$	An operator of the HSM
NXD	Non Exportable and Temporary Data Storage
ODS	Operator Data Storage
OUT	External output of HSM.
KDS	Key Data Storage
CRL	Certificate Revocation List.

Table 7. Description of all Objects and Message Parts of the Protocols

Message	**Description**
M	Size of the administrators group.
N	Minimum amount of administrators to reconstruct a shared secret.
I	Any principal part of a group.
Kr_{HSM}	Private key of the HSM.
Ku_{HSM}	Public key of the HSM.
Kr_{ADM_I}	Private key of one Administrator.
Ku_{ADM_I}	Public key of one Administrator.
ADM_I	Identification for one Administrator.
Ks_{ADM}	Symmetric key used for ciphering Kr_{HSM}.
Ks_{ADM_I}	Shared part of Ks_{ADM} belonging to one Administrator.
N_I	A random value just used once (Nonce).
K	Size of the operators group.
L	Minimum amount of operators to reconstruct a shared secret.
$OPER_{ID}$	Identifier of a operators group.
Kr_{OPER_I}	Private key of one operator.
Ku_{OPER_I}	Public key of one operator.
$OPER_{ID_I}$	Identification for one operator.
Ks_{OPER}	Symmetric key for ciphering private keys owned by operators group.
Ks_{OPER_I}	Shared part of Ks_{OPER} belonging to one Operator.
Ks_{OPER*}	Non exportable and temporary operand that enables administrative operations to an operator group.
KEY_ID	Identifier for the key being generated or being used.
KEY_PARAM	Parameters for the key being generated.
Kr_{APP}	Application protected private key.
Ku_{APP}	Public key related with Kr_{APP}.
KEY_POL	Policies for a key being used.