Key management system with HSM

Vítor André Lima Neto
202103153

June 24, 2022

1 Brief Introduction

A key management system is a system for the management of cryptographic keys with the objective of dealing with operations such as the creation and storage of these keys. In this specific project we create a key management system based on hardware security models.

An hardware security model is a physical hardware component that can store cryptographic keys, and can perform cryptographic operations such as encryption, decryption, signatures and others. The security of such a component is based on the isolation of the cryptographic operations when done by such a device. Rather than have cryptographic operations being performed on possibly non-secure or compromised computers they are done in physically secure modules on isolated environments.

The created key management system has three main components on which it bases it's security, the Domain as a key structure, the operator and the hardware security model.

Github project

2 Security

Three main elements comprise and characterize the security profile of a key management system based in hardware security models:

1. These are the hardware security model, which provides a secure environment and entity.
2. The operator an entity that can be honest or dishonest, they serve as a key policy manager, the system should be created in such a way that a dishonest operator can not interfere or compromise the system.
3. The key structure element that allows a community, or cluster, of hardware security models to operate with inherited design trustworthy content and minimizing the need for intercommunication, this domain keys should also be secure enough such as that being stored on the client or outside a secure environment or where to be compromised (where compromised would mean non-authorized access by a third party, not cryptographic compromise) it would still be secure and the destruction and recreation of a new Domain would not large scale affect the key management system.

2.1 Hardware security model

The security of the hardware security model is the capacity to perform security operations on an isolated hardware component, preventing the possibility of system compromise as it could happen in other alternative models. It provides an isolated environment to use cryptographic keys to perform cryptographic operations without fear of side channel attacks. Due to the limited usage in terms of space in these hardware models it means we cant store all cryptographic keys in the safety of the hardware security model so key structures must be created and managed to take advantage of the hardware security model securities characteristics.

2.2 Operators

An operator is an entity in this key management system, an operator can be honest or dishonest, an operator validates a Trust authenticity, creating a signature if valid, allowing an *hsm* to sign the Trust as a authenticity element. A dishonest operator might create or sign a compromised Trust. A honest operator would only sign a authentic Trust.

2.3 Key structures

The base key structure to take advantage of a key management system with hardware security models is the Domain, which stores a Trust, a Domain keys and a signature of these elements, as it can be seen in figure 3. This key structure element allows a cryptographic key to be stored in a way that its functionality and security is based on the hardware security models, so any access to this structure, authorized or not, does not compromise the cryptographic key in it unless one of the hardware security models on the system is also compromised. This means that this structure could be stored anywhere, from a dedicated system with policies control, our the client private equipment.

2.3.1 Domain keys

The Domain keys, figure 1, is the structure on which we base our cryptographic security and make use of the hardware security module advantages.

The domain keys is a key structure that stores cryptographic keys by taking advantage of asymmetric encryption. The Domain keys as the objective of storing and protecting a master key, mk, as an encryption, this resulting structure was called a token, an encrypted cryptographic key.

This mk is encrypted by an arbitrary symmetric key, k, that should be generated at creation time by the hardware security model where this process it taking place, this k must now be protected as any access to it would allow the decryption and usage of the master key, so this is now done by taking advantage of the hardware security models in the system, we assume that these models can store a limited amount of information, and for this system that information is a key par for asymmetric encryption, with that in mind we can now encrypt the generated k with all valid *hsm* public keys in our system, creating n amount of tokens of that same k, where n is the number of authentic and trustworthy hardware security models in our system.

With this structure all *hsm*, which public keys were used to encrypt the generated key, can now decrypt it and further decrypt the master key that was encrypted with such key, allowing the hardware security model to perform cryptographic operations with the master key.

More formally: $E(Pki, k)$, where E function is the encryption of k using Pki, and it is done for every Pki, where Pki are the *hsm* public keys in Trust, and also $E(k, mk)$, where k and mk are the Domain specific generated cryptographic keys.

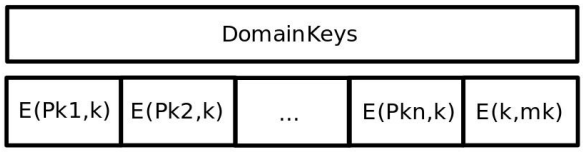

Figure 1: Domain keys.

2.3.2 Trust

The trust, figure 2, is the structure that allows policy control of the domain, this structure is composed of six elements, and identifying number, the public keys belonging to the hardware security model and the operators, these public keys are the ones allowed to operate in the Domain to which the Trust is aggregated, a quorum value which defines a minimum value of operators in the trust needed to validate the same trust, the hash of the predecessor hash and a signature, created by an *hsm*, to authenticate this same Trust.

The Trust signature is created by an hardware security model to which it's public key is also in the Trust, for the hardware security model to sign the Trust it must also receive a number, equal or

more than the Quorum, of signatures of the operators that also have their public key in the Trust, these operators create their signature after validating the authenticity of the elements in the Trust, such as the hardware security models public keys, if a dishonest operator falsifies or alters a trust with compromised or non-trustworthy elements, such as a fake/compromised hardware security model public key, then a honest operator would not sign the Trust.

The Quorum value is the minimum number of operator signatures of the Trust required so that a *hsm* can itself sign the Trust, for example if we have ten operators in the Trust, and a Quorum value of eight, that means that we need eight operators to verify the Trust content authenticity and sign it, allowing the *hsm* to also sign it for later verification and authentication when using a Domain for cryptographic operations, this means that we need at least three honest operators in the Trust to prevent a compromised Trust from being created, if seven operators agreed to create a Trust with compromised public keys, then the other three honest operators, when verifying the Trust will notice the non-authentic public keys and not sign the Trust, not creating the minimum number of eight signatures that would allow the *hsm* to sign the Trust.

Another alternative to this situation, and the which that would make it so that you would not want to have a Quorum number equal to the operator amount in the Trust, is that a dishonest operator might not sign an honest Trust, preventing the Quorum number from being attained and the Trust from being signed and used, in an Quorum number of eight it means that you could have up to two dishonest operators and they would not be successful in an attempt to denial the Trust utility and usage.

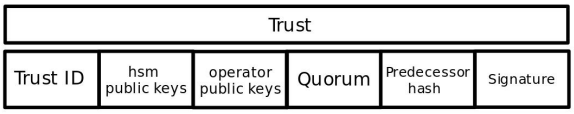

Figure 2: Trust.

2.3.3 Domain

The Domain is the combination/aggregation of a Trust with a Domain keys, where the Domain keys is created with the Trust information, such as the *hsm* public keys. This Domain is also signed by an *hsm* when created, as to validate it's authenticity and prevent falsification. All hardware security models which public keys are in the Trust of the Domain, then all this hardware security models have the ability to validate the signatures of the Trust itself and of the Domain, they all can also operate the Domain keys and perform cryptographic operations.

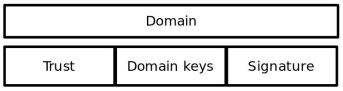

Figure 3: Domain.

3 Implementation

The described key management system was implemented in java programming language. The architecture visualized in the figure 4 is the one implemented, four main entities can be identified, the Client, the Operator, the hardware security models and the Server, within the server a manager is used to perform actions, this manager keeps track of all relevant information. such as the current public keys in service, and the available Domains and Trusts, this general manager also keeps track of the hardware security models, this devices are entities within our manager.

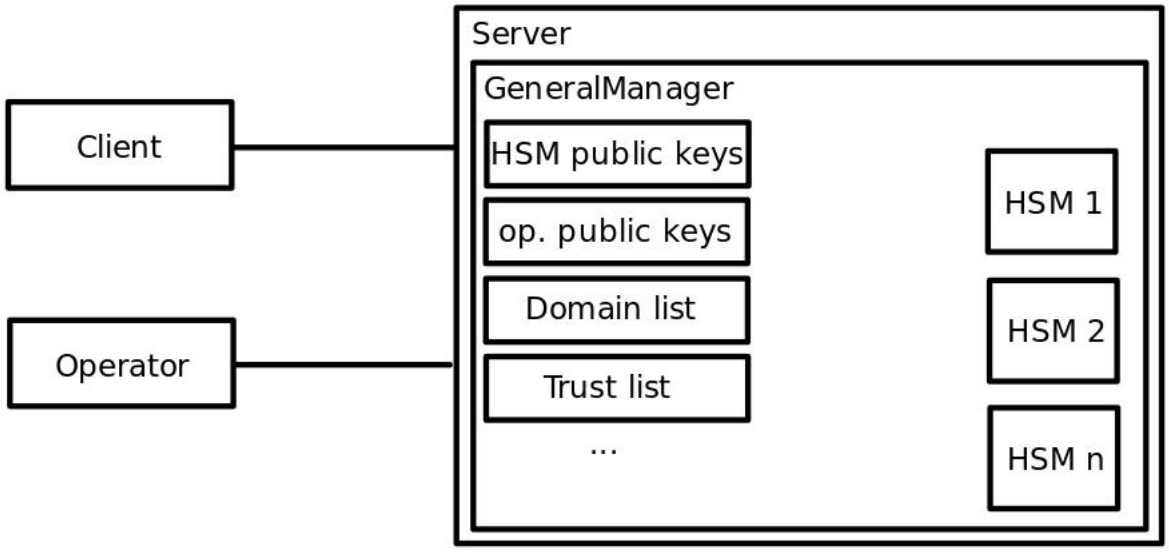

Figure 4: The distributed system architecture.

3.1 Hardware security model

The hardware security model, figure 5, performs by inheritance cryptographic operations, such as encryption, decryption, signatures, and others, this operations are done by resorting to RSA and AES. The hardware security models can then perform more advance operations such as Trust and Domain creating, and then operations with the Domain such as encryption and decryption by unwrapping the relevant keys for usage. The hardware security model can also verify the integrity and authenticity of the Domains and Trusts that it receives for operations. Each hardware security model only stores its key pair for asymmetric encryption.

The *hsm* can create new Trusts, this Trusts are unsigned and with content indicated by an operators, such as what operators and *hsm* should be part of the Trust, and also a quorum value. This Trust can then be signed by the *hsm* but it only does so when it receives a number, equal or more than the Quorum, of valid signatures by the operators who are part of the Trust, if at any point the signatures fails or the number of signatures received are not enough then the *hsm* does not sign the Trust.

For a Domain to be created by an *hsm* a Trust must be given, the *hsm* must be part of that Trust, and the Trust signature must be valid. The creation of the Domain implies the creating of the Domain keys, a symmetric key is generated to encrypt a generated master key, this master key can also be a key pair, and a token is created; the *hsm* public keys in the Trust are then used to encrypt the generated symmetric key used to encrypt the master key, creating an n set of tokens, where n is the number of total *hsm* public keys in the Trust. After the Domain keys is created an signature of the Domain keys and Trust is created by the *hsm*, aggregating the three components to build a Domain; this Domain can then be used for cryptographic operations such as encryption or decryption by the unwrapping process allowing access to the encrypted master key, this process can only be done by an *hsm* which public keys is in the Domain Trust.

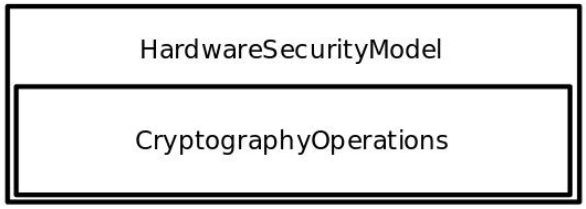

Figure 5: HSM model.

3.2 Operator

The operator is the entity that has the ability to create Domains and Trust, and asks the hardware security models to sign such structures when possible (the *hsm* might not sign the structure if the Quorum value of operator Trust signatures is not achieved). The operator can visualize all the elements in the system, such as existing Trusts, signed and not signed, existing Domains, hardware security modules in service, and others. In this key management service the Operators themselves have a Domain to perform cryptographic operations in the system such as signing the Trusts, this Domain stores a key pair, where the private key is encrypted. The operator can be honest or dishonest, and taking into account the descriptions of the key structures above, the system tolerance for dishonest operators relates to the Quorum value, where you can not have a number of dishonest operators in a Trust equal to the Quorum value. The operator connects to the server using method remote invocation, the authentication of the operator is assumed.

In sum the operator can perform the following actions:

1. View the unsigned Trusts in the system
2. View the signed Trusts in the system
3. View Trusts of which it is part
4. View the domains in the system
5. View operator public keys in the system
6. View *hsm* public keys in the system
7. View operator signatures of unsigned Trusts
8. Create unsigned Trust
9. Sign unsigned Trust
10. Sign unsigned Trust with *hsm*
11. Verify Domain signatures
12. Verify Trust signatures
13. Create new *hsm*

3.3 Client

The client is the entities that takes advantage and uses the key management service, the authentication of the client is assumed. The client claims a Domain and by providing a file it performs encryption or decryption operations.

In sum the client can perform the following actions:

1. Encrypt file with Domain
2. Decrypt file with Domain

3.4 Server

The server contains the manager service which provides the operations, it also stores and contains the relevant information, since the operators have Domains, which it uses for its operations such as signatures, the first start of the system must create a beginning Trust to create the Domains for the operators, this starting Trust only has the *hsm* public keys that exist at the start, that Trust also has no operator public keys. The interaction with the Server is done with java remote method invocation.