République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Blida 1
Faculté des Sciences
Département de l'informatique

Mémoire de fin d'études en vue de l'obtention du Diplôme

MASTER EN INFORMATIQUE

Option : Sécurité de Système d'information

Thème:

Gestion de clés de chiffrement en tant que service de sécurité de confiance basé sur les TPM dans le Cloud deComputing

Présenté par :Melle SAIDIA SIHAM et Mr MEKHLOUF Abderrahmane

Le : 28/06/2018

Devant les jurysEncadré par :

- Dr :Gheribi (Président)
- Dr :Djedar (Jury)
- Dr :N.Boustia (USDB)

(Encadrant)

- Dr :O.Nouali (CERIST)
- Mr :S. Fehis (ENSSA-Alger)

2017/2018

Dédicace

À nos très chers parents,

Aucun mot, aucune dédicace ne peut exprimer notre respect, notre considération et l'amour éternel pour les sacrifices que vous avez consentis pour notre instruction Et notre bienêtre.

Votre générosité et votre bonneté ont toujours été un exemple pour nous tous.

Trouvez en ce travail le fruit de votre dévouement de votre patience et l'expression de notre gratitude et profond amour

Je t'aime papa je t'aime maman et sachez que je suis très reconnaissant

À toute notre chère famille,

À nos professeurs,

À nos chers amis,

À nos chers collèges,

À tous ceux qui m'aiment,

À tous ceux que j'aime,

À tous ceux qui nous ont aidé de près ou de loin,

Je dédie ce travail avec hommage

Remerciements

Avant tout, nous remercions Dieu le très haut qui nous a donné le courage et la volonté de réaliser ce modeste travail.

« (CELUI QUI NE REMERCIE PAS LES GENS, NE REMERCIE PAS ALLAH.)»

[Authentique Hadith]

Nous remercions avant tout le Dieu tout puissant d'avoir veiller sincèrement sur notre état de santé pendant ces Cinq années de notre cursus universitaire en nous donnant ainsi la force et le courage de bien mener à terme ce modeste travail

Nous remercions Dr. Nouali Omar Directeur de Recherche CERIST et Mr Fehis Saad de nous avoir donné l'opportunité de faire ce travail ainsi que pour leur conseils judicieux qui nous ont permis d'avancer dans notre travail.

Que Madame N. Boustia , professeur De l'Informatique à l'université «Saad Dahleb », trouve ici le témoignage de notre profonde reconnaissance. Ses encouragements, et surtout ses critiques, Sa sensibilisation, ont largement contribué à l'accomplissement de notre travail.. Nous la remercions infiniment de nous avoir toujours poussés vers l'avant.

On tient également à remercier « TOUS » les Messieurs et dames, nos professeurs qui nous ont enseigné durant deux ans master en sécurité de système D'information, pour leurs précieux conseils et ses orientations,

Enfin, merci à nos familles (nos Chères mères et nos père) pour le soutien et l'encouragement qu'ils nous ont apporté tout au long de ce travail.

Résumé

L'externalisation des applications informatiques vers le Cloud-Computing, offre plusieurs avantages pour les entreprises en matière de ressources et disponibilité. En effet, les utilisateurs mobiles utilisent ces applications sans traverser leurs réseaux d'entreprise. Cependant, les ressources partagées causent de nombreux problèmes de sécurité, faisant que la sécurité est l'une des principaux obstacles à l'adoption des services de Cloud-Computing. A cet effet, le vrai challenge, est comment créer la confiance (the trust) entre le client et le fournisseur de services dans le Cloud Computing.

Dans ce travail, on s'intéresse, à offrir aux clients des composants softwares de confiance, qui peuvent assurer la sécurité en tant que service (Security as a Service, SecaaS). Le composant est utilisé en tant que plateforme de confiance d'un système de gestion des clés de chiffrement (Cryptographic Key Management System CKMS). A cet effet, le défi est la sécurisation du composant ainsi que son fonctionnement dans un environnement totalement virtualisé & multi-locataire. À savoir la création des clés (chiffrement / signature) dans un contexte de confiance en toute sécurité. Pour cela, le module de plateforme de confiance (Trusted platform module TPM]) en tant que hardware, est considéré comme base de confiance. A cet effet ; l'objectif de ce travail, est l'utilisation du TPM comme source de confiance de création des clés de chiffrement pour le CKMS en tant que SecaaS.

Mot clés : *Cloud-Computing ,Security as a Service (SecaaS) , Crypto graphic Key Management System(CKMS), Trusted platform module (TPM).*

ملخص :

ان تفتح التطبيقات المعلوماتية على استخدام السحابية المعلوماتية فتح العديد من المحاسن و التسهيلات للمؤسسات وذلك بتوفير الموارد بأقل الأسعار و في كل الأوقات و ذلك عن طريق استعمال الأجهزة النقالة و الشبكة فقط.

و بما أن هذه الموارد مشتركة فتحت العديد من المشاكل الأمنية وعليه تطرح الإشكالية المتمثلة في كيفية تحقيق الأمن والثقة بين مستخدمي ومقدمي الخدمات في السحابة المعلوماتية.

في هذا المشروع سوف نقوم باستخدام أجهزة تقوم بضمان الأمن على شكل خدمات (Secaas) يتم استخدام هذا الجهاز على شكل منصة موثقة لنظام إدارة المفاتيح التشفير (CKMS) وذلك لضمان أمن موارد المؤسسة التي تعمل في بيئة افتراضية ومتعددة المواقع.

*يتم تشكيل مفاتيح (مشفرة، موقعة) بطريقة موثقة و مؤمنة عن طريق وحدة نظام الأساسي الموثوق (**TPM**) كأجهزة ثقة، و الهدف من كل هذا أن نقوم باستعمال **TPM** كمصدر لثقة أثناء صنع مفاتيح التشفير التي يديرها CKMS على شكل خدمة مؤمنة .*

الكلمات المفتاحية : *منصة موثقة لنظام إدارة المفاتيح التشفير **(CKMS)** ، الأمن على شكل خدمات **(Secaas)** ، وحدة نظام الأساسي الموثوق **(TPM)***

Abstract :

The Outsourcing IT applications to Cloud-Computing offers many advantages in terms of resources and availability for the business (industrie and factories). In fact, mobile users use these applications without crossing their corporate networks. However; shared resources cause many security issues,doing so security is one of the principle barriers to cloud-computing services adoption .for this purpose, the real challenge is how to create trust between the customer and the service provider in cloud computing.

In this work, we are interested in offering customers trusted software components that can provide security as a service (SecaaS) .The component is used as a trusted platform for Cryptographic Key Management System (CKMS) The challenge is to secure the component and its functioning in a fully virtualized & multi-tenant environment. Namely the creation of keys (encryption / signature) in a context of trust and security. For this purpose, the trusted platform module (TPM) as hardware is considered as a basis of trust. the purpose of this work, is the use of TPM as a trusted source of creation of encryption keys for CKMS as SecaaS.

Keywords : Cloud-Computing, Security as a Service (SecaaS) , Crypto graphic Key Management System(CKMS), Trusted platform module (TPM).

Table des matières

Liste des figures

Liste des Tableaux :

Liste des abréviations

ACS: Access Control System .

ASP: Application Service Provider

API: Application Programming Interface.

CA : autorité de certification .

CKMS : Cryptographic Key Management System .

CRT : Le mode compteur

DNS : Domain Name System .

EK : Clé D'approbation..

ECC : Elliptic Curve Cryptography

EPP : Endpoint Protection Platform.

IDS : Intrusion Detection System .

IPS : intrusion prevention system

KDF : Fonction de dérivation de clé

KDC : centre de distribution des clés

NIST : National Institute of Standards and Technology

OS : Operating System

PCI: Industrie des cartes de paiement.

PHP : *Hypertext Preprocessor*

RA : autorité d'enregistrement

RNG : Génération de nombre aléatoire

RSA : Rivest Shamir Adleman

RTM : Racine de Confiance pour la Mesure

SHA : Secure Hash Algorithm

SOA : Architecture Orientée Services

SQL : Standard Query Language

TCG : Trusted Computing Group

TCB : Trusted Computing Base

TPM: Trusted Platform Module

VM : machine virtual.

WAF : Web Application Firewall .

VMM: virtual manager Machine.

VTPM: virtual Trusted Platform Module

Introduction Général

Au fur et à mesure que les systèmes informatiques évoluent, la demande en quantité d'espace de stockage, de confidentialité et de simplicité dans le travail va en grandissant. Il y a quelques années, les espaces de stockage réduits, les lignes de commandes et les systèmes complexes étaient le quotidien des employés d'entreprise.

Les entreprises modernes traitent de grandes quantités d'informations aussi nombreuses que variées. Ainsi, elles ont besoin de grande capacité de stockage et une puissance de calcul élevée. Les ressources matérielles et logicielles nécessaires n'étant pas à la portée de toutes les entreprises, « le Cloud Computing » est une solution pour résoudre ce problème.

Le Cloud Computing est maintenant le fondement le plus utilisé d'Internet : Email, moteurs de recherche, réseaux sociaux, médias en streaming, et d'autres services sont désormais hébergés dans "le Cloud ". Le Cloud Computing a un coût réduit et une commodité accrue, l'accessibilité et la centralisation du Cloud Computing crée également de nouvelles opportunités pour les failles de sécurité. [1] À cet effet le vrai challenge est comment créer la confiance entre le client et le fournisseur de service dans le Cloud Computing ?

Dans ce travail, on s'intéresse à offrir aux clients des composants softwares de confiance, qui peuvent assurer la sécurité en tant que service (*Security as a Service, SecaaS*) dans le Clous Computing. Pour cela e composant est utilisé en tant que plateforme de confiance d'un système de gestion des clés de chiffrement (*Cryptographic Key Management System CKMS*). Qui gérer les clés de chiffrements A cet effet, le défi est la sécurisation du composant ainsi que son fonctionnement dans un environnement totalement virtualisé & multi-locataire. A savoir la création des clés (*chiffrement / signature*) dans un contexte de confiance en toute sécurité. Pour cela, le module de plateforme de confiance (*Trusted platform module TPM*) en tant que hardware est considéré comme base de confiance et source sûre de génération des clés. Telle que cette pause cryptographique créer les déférents types de clés avec déférentes Algorithmes A cet effet, l'objectif de ce travail, est d'offrir des services de type *CKMS* en tant que SecaaS dans le cloud basé sur le TPM.

Le service SecaaS, utilise le TPM pour la création des clés, son chiffrement et sa signature. En effet, le CKMS lui-même n'est qu'un intermédiaire entre l'utilisateur (*demandeur de la clé*) et le TPM (*générateur de la clé*) sans déchiffrement.

Ce document est structuré de cinq chapitres :

Le premier chapitre est un background sur le Cloud-Computing, ses service en général, et les services de type sécurité (*SecaaS*) avec leurs challenges de sécurité notamment le problème de confiance pour l'adoption des services offertes par le Cloud-Computing.

Le deuxième chapitre est divisé en deux parties : Une pour la cryptographie et l'autre présente une étude et analyse de l'architecture et le fonctionnement du CKMS.

Le troisième chapitre décrit l'architecture et le fonctionnement de TPM, ses capabilités dans la gestion des clés de chiffrement/signature.

Dans le quatrième chapitre, nous allons présenter l'approche du CKMS en tant SecaaS basé sur le TPM, la conception et la modélisation (*utilisation d'UML*) de la solution.

Dans le cinquième chapitre, nous présentons la description de l'application, des tests & résultats et validation de la solution.

En fin nous concluons par une conclusion générale avec des perspectives de notre travail.

CHAPITRE I: CLOUD COMPUTING

Chapitre I: Cloud Computing

I.1 -Introduction

Le Cloud Computing est un concept qui n'est pas récent dans le domaine de l'informatique. La première énonciation de ce concept date de 1960. Il envisageait à cette époque que les matériels, équipements, ou installations informatiques pouvaient être délivrés aux utilisateurs sous forme de services. Ce concept a évolué dans le temps et est actuellement à une étape de concrétisation.

Dans ce chapitre nous allons présenter les notions fondamentales de Cloud Computing, la technologie qui la constitue.

I.2 –Historique de Cloud Computing

Le Cloud Computing possédé différentes perceptions chez différentes personnes on peut citer les suivantes :

Pour certains, il se réfère à l'accès au logiciel et le stockage des données dans la représentation "Cloud" Internet ou un réseau et en utilisant les services associés.

Pour les autres, on le voit juste comme une modernisation du modèle de partage du temps qui était largement utilisé dans les années 1960 avant l'avènement de l'informatique à moindre coût. Ces développements ont fini par évoluer vers le client/serveur et à l'ordinateur personnel, qui a placé de grandes quantités de calcul sur les ordinateurs des gens et orthographié la disparition des systèmes de partage du temps [1] .

D'autres pensent que la discipline n'est apparue que plus tard, dans les années 2000, telle une solution à un problème rencontré par **Amazon** : le surdimensionnement de son parc de serveurs hors période de fêtes. Pour rentabiliser ces machines, il décida de louer ses serveurs à d'autres entreprises à la demande, et que la première personne à employer l'expression de Cloud Computing fut le professeur **Ramnath Chellappa** de l'université du Texas à Austin en 1997.Et **Salesforces,** en 1999, fut le premier, à transformer ce concept en business avec le logiciel de gestion de la relation client éponyme, **Amazon** lui emboîte le pas en 2002.

- C'est sans doute grâce à **IBM** que tout le monde en parle aujourd'hui. En effet, en 2007, il décida de faire de ce concept l'une des lignes de force de sa stratégie. [2]

I.3 – Définition de Cloud Computing

L'image dans la figure 1 représente un exemple d'une Cloud Computing

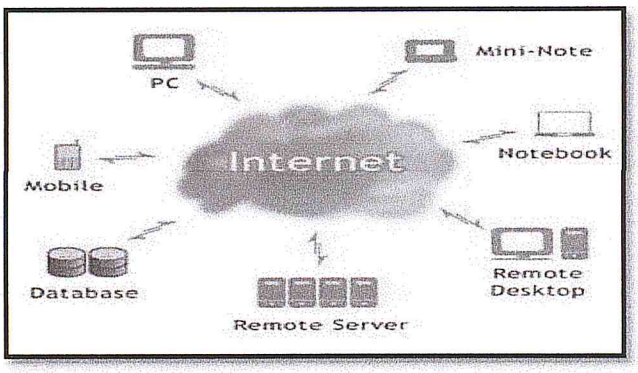


Figure 1 : Cloud Computing *[3]*

Il existe plusieurs définitions de Cloud Computing tels que :

1- Le Cloud Computing traduit de l'anglais « informatique dans les nuages » est devenu un mot à la mode populaire, littéralement l'informatique dans les nuages est un concept qui consiste à déporter sur des serveurs distants des stockages et des traitements informatiques traditionnellement localisés sur des serveurs locaux ou sur le poste de l'utilisateur.

Il consiste à proposer des services informatiques sous forme de service à la demande, accessibles de n'importe où, n'importe quand et par n'importe qui, grâce à un système d'identification, via un PC et une connexion à Internet Cette définition est loin d'être simple à comprendre, toute fois l'idée principale à retenir est que le Cloud n'est pas un ensemble de technologies, mais un modèle de fourniture, de gestion et de consommation de services et de ressources informatiques. [4]

2- Dans une présentation d'octobre 2009 intitulée « Utiliser efficacement et en toute sécurité le Cloud Computing Paradigmes 3 » par Peter Mell et Tim Grance de laboratoire de National Institute de Standards et Technologie (*NIST*), le Cloud Computing est défini comme suite :

Le Cloud Computing est un modèle permettant un accès réseau pratique et à la demande à un pool partagé de ressources informatiques fiables (*par exemple : des réseaux, serveurs, stockage, applications, services*) pouvant être rapidement provisionnés et libérés avec un minimum d'effort de gestion du consommateur ou d'interaction avec le fournisseur de services. [1]

I.4- Eléments de Cloud Computing

On peut définir les éléments de Cloud Computing comme suit :

I.4.1- La virtualisation

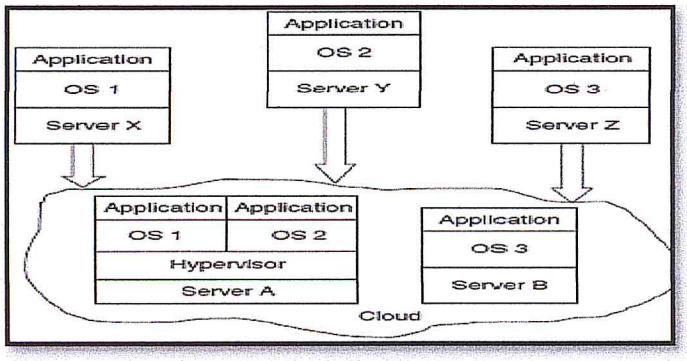


Figure 2 : Exemple de virtualisation *[5]*

Se définit comme l'ensemble des techniques matérielles et/ou logiciels qui permettent de faire fonctionner sur une seule machine, plusieurs systèmes d'exploitation, appelées machines virtuelles (*VM*), ou encore OS invitée [6].

La virtualisation des serveurs permet un bien de plus grande modularité dans la répartition des charges et la reconfiguration des serveurs en cas d'évolution ou de défaillance momentanée.

I.4.2- Centre de Calcul (Datacenter)

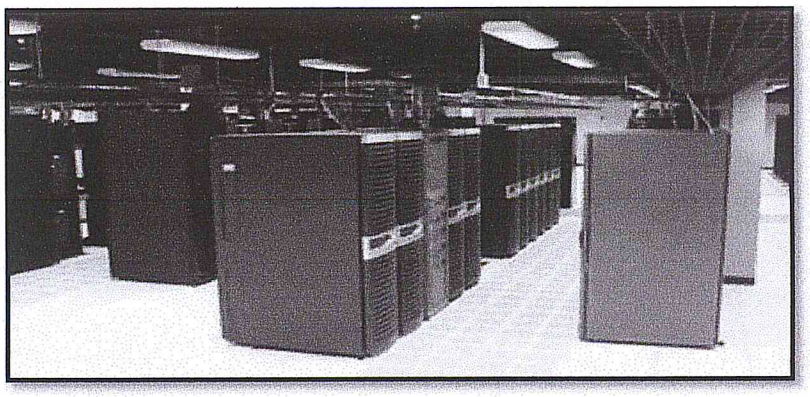


Figure 3 : Datacenter *[7]*

Un centre de traitement de données (*data centre en anglais*) est un site physique sur lequel se trouvent regroupés des équipements constituants le système d'information de l'entreprise (*mainframes, serveurs, baies de stockage, équipements réseaux et de télécommunications, etc.*), il peut être interne et/ou externe à l'entreprise, exploité ou non avec le soutien de prestataires, il

comprend en général un contrôle sur l'environnement (*climatisation, système de prévention contre l'incendie, etc.*), une alimentation d'urgence et redondante, ainsi qu'une sécurité physique élevée. Cette infrastructure peut être propre à une entreprise et utilisé par elle seule ou à des fins commerciales. Ainsi, des particuliers ou des entreprises peuvent venir y stocker leurs données suivant des modalités bien définies [6].

I.4.3- Plateforme Collaborative

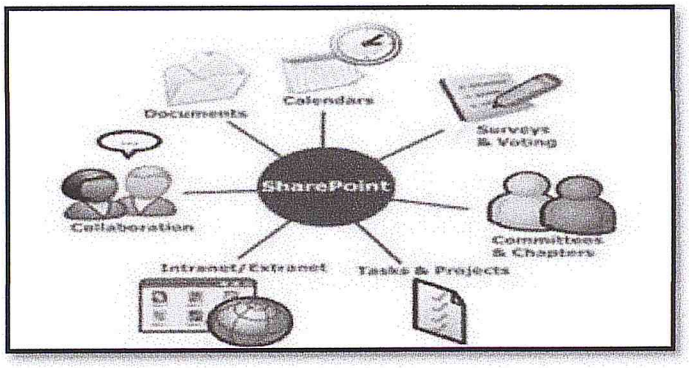


Figure 4 : Plateforme collaborative *[8]*

Une plate-forme de travail collaborative est un espace de travail virtuel. C'est un site qui centralise tous les outils liés à la conduite d'un projet et les met à disposition des acteurs. L'objectif du travail collaboratif est de faciliter et d'optimiser la communication entre les individus dans le cadre du travail ou d'une tâche.

Les plates-formes collaboratives intègrent généralement les éléments suivants : Des outils informatiques ; des guides ou méthodes de travail en groupe pour améliorer la communication, la production, la coordination, Un service de messagerie ; Un système de partage des ressources et des fichiers . Des outils de type forum, pages de discussions ; Un trombinoscope, ou annuaire des profils des utilisateurs, des groupes, par projet ou par thématique ; un calendrier. [8]

I.5 -Types de Cloud Computing

Le Cloud Computing peut être répertorie en trois catégories :

I.5.1- Cloud Publique

L'infrastructure de Cloud est mise à la disposition du public ou un grand groupe industriel et appartient à une organisation vendant des services de Cloud Computing. [9]

Examples: Amazon Elastic Compute Cloud (EC2), Sun Cloud, IBM's Blue Cloud, Google AppEngine et Windows Azure Services Platform. [1]

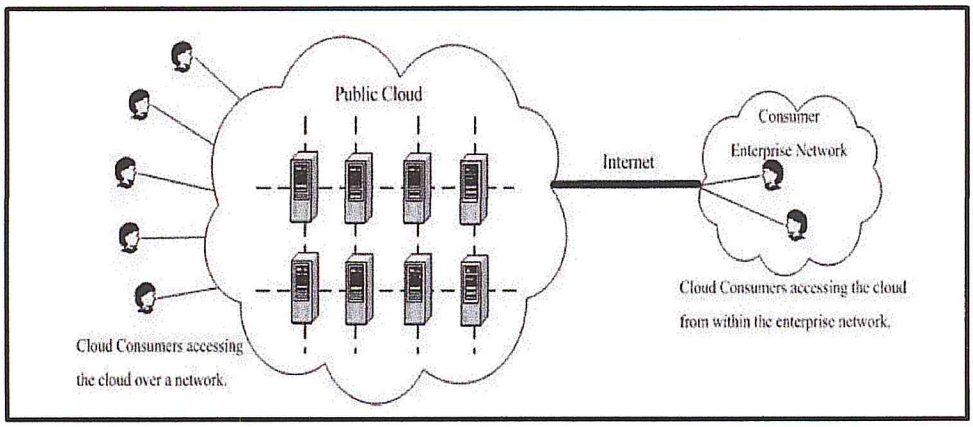


Figure 5 : Cloud public *[4]*

I.5.2- Cloud Privé

L'infrastructure de Cloud est exploitée uniquement pour une organisation. Elle peut être gérée par l'organisation ou un tiers et peut exister sur prémisse ou hors prémisse. [9]

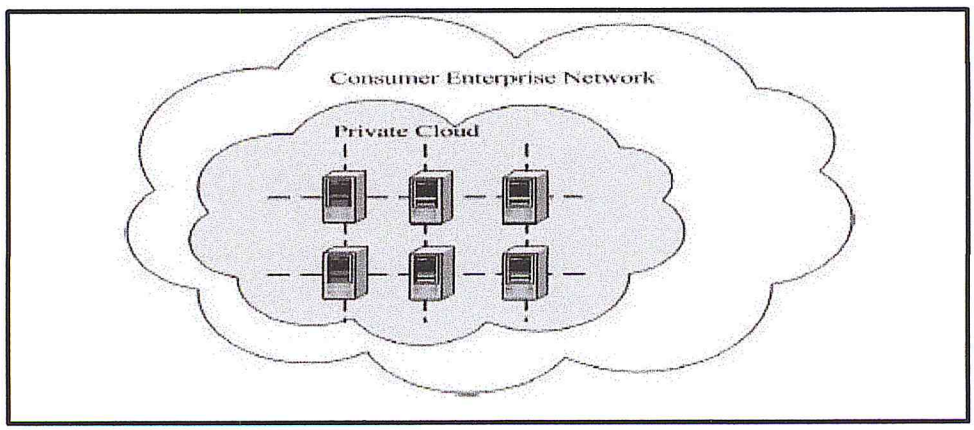


Figure 6 : Cloud privé *[4]*

I.5.3- Cloud Hybrid

Un Cloud Hybride est l'utilisation de plusieurs Cloud, publics ou privés amenés à « Coopérer », à partager entre eux applications et données. [9]

I.6- Services de Cloud Computing

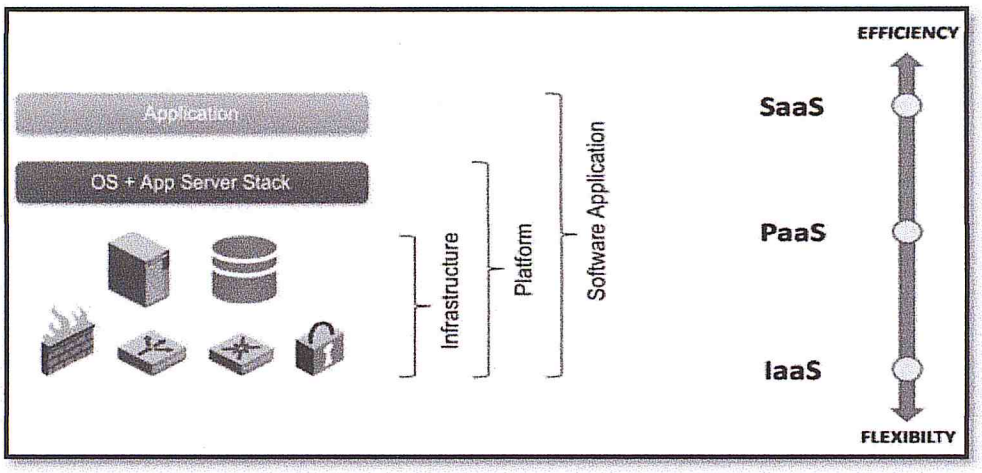


Figure 7 : L'architecteur de Cloud Computing *[10]*

La partie visible du Cloud Computing se décompose en 3 parties qui sont :

I.6.1- Infrastructure as a Service (*IaaS*)

Il s'agit de la mise à disposition, à la demande, de ressources d'infrastructures dont la plus grande partie est localisée à distance dans des Datacenter. L'IaaS permet l'accès aux serveurs et à leurs configurations pour les administrateurs de l'entreprise.

Le client a la possibilité de louer des clusters, de la mémoire ou du stockage de données. Le coût est directement lié au taux d'occupation. Une analogie peut être faîte avec le mode d'utilisation des industries des commodités (*électricité, eau, gaz*) ou des Télécommunications.

- **Avantage** : Grande flexibilité, contrôle total des systèmes (*administration à distance par SSH ou Remote Desktop, RDP*), qui permet d'installer tout type de logiciel métier.
- **Inconvénient** : Besoin d'administrateurs système comme pour les solutions de serveurs classiques sur site. Les cibles sont les responsables d'infrastructures informatiques. Amazon EC2 est le principal acteur qui propose ce genre d'infrastructures. Eucalyptus est un exemple d'infrastructure. [4]

I.6.2- Platform as a Service (*PaaS*)

Il s'agit des plateformes du nuage, regroupant principalement les serveurs mutualisés et leurs systèmes d'exploitation. En plus de pouvoir délivrer des logiciels en mode SaaS, le PaaS dispose d'environnements spécialisés au développement comprenant les langages, les outils et les modules nécessaires.

L'avantage est que ces environnements sont hébergés par un prestataire basé à l'extérieur de l'entreprise, ce qui permet de ne disposer d'aucune infrastructure et de personnel de maintenance et donc de pouvoir se consacrer au développement.

- **Avantage :** Le déploiement est automatisé, pas de logiciel supplémentaire à acheter ou à installer.
- **Inconvénient :** Limitation à une ou deux technologies. [4]

Exemple : Windows Azure de Microsoft, AppEngine de Google, Force.com de Salesforce. Chaque fournisseur de PaaS propose des environnements de développement différents, Google AppEngine se limite à Java et Python, tandis Windows Azure permet de travailler avec les langages .NET, PHP, Python, Ruby et Java**.** [11]

I.6.3- Software as a Service (*SaaS*)

Concept consistant à proposer un abonnement à un logiciel plutôt que l'achat d'une licence. On oublie donc le modèle client-serveur et aucune application n'est installée sur l'ordinateur, elles sont directement utilisables via le navigateur Web. L'utilisation reste transparente pour les utilisateurs, qui ne se soucient ni de la plateforme, ni du matériel, qui sont mutualisés avec d'autres entreprises. Le SaaS remplace l'ASP (*application service provider en anglais ou ASP*), qui est une entreprise qui fournit des logiciels ou des services informatiques à ses clients au travers d'un réseau. Deux principales différences avec l'ASP traditionnel sont : Qu'une simple interface web est utilisée côté client dans tous les cas (pas de client lourd), et que le SaaS propose une seule instance de logiciels qui évolue indépendamment des clients. Avec l'arrivé du Haut débit, les logiciel en mode SaaS deviennent utilisables sans problèmes.

- **Avantages :** plus d'installation, plus de mise à jour (*elles sont continuées chez le fournisseur*), plus de migration de données etc. Paiement à l'usage. Test de nouveaux logiciels avec facilité.
- **Inconvénients :** limitation par définition au logiciel proposé. Pas de contrôle sur le stockage et la sécurisation des données associées au logiciel. Réactivité des applications Web pas toujours idéale.

Les cibles sont les utilisateurs finaux. On retrouvera des entreprises comme Sales Force ou Diva (projet lancé par une équipe de l'école d'ingénieur EPITECH). [4]

Exemple: Google docs **http://docs.google.com**

Le modèle Cloud Computing lui-même induit certains risques liés à la sécurité, il ouvre de nouvelles opportunités pour obtenir une sécurité innovante des solutions techniquement et économiquement flexibles de manière à faire face aux demandes de sécurité croissantes.

L'externalisation de la sécurité selon les principes du SaaS est appelé Security as a Service (*SecaaS*).

I.7- Sécurité en tant que service (*SecaaS*)

I.7.1-Définition

Security as a Service (*SecaaS*) est une architecture orientée services (*SOA*) qui gère la sécurité du Cloud Computing. Le principe de base de SecaaS est de donner aux utilisateurs de Cloud plus de contrôle sur le processus de sécurisation de leurs applications et des données. SecaaS cible tous les niveaux de Cloud Computing.

SecaaS protège les ressources de l'utilisateur et du fournisseur. Étant donné un Cloud, SecaaS est appliqué aux niveaux SaaS, PaaS et IaaS. À niveau SaaS, les services de sécurité protègent les données des utilisateurs, ainsi que logiciel du fournisseur. Au niveau PaaS, les services de sécurité protègent les applications utilisateur, ainsi que la plate-forme fournisseur. Au Niveau IaaS, les services de sécurité protègent les machines virtuelles des utilisateurs, ainsi que l'infrastructure du fournisseur [12].

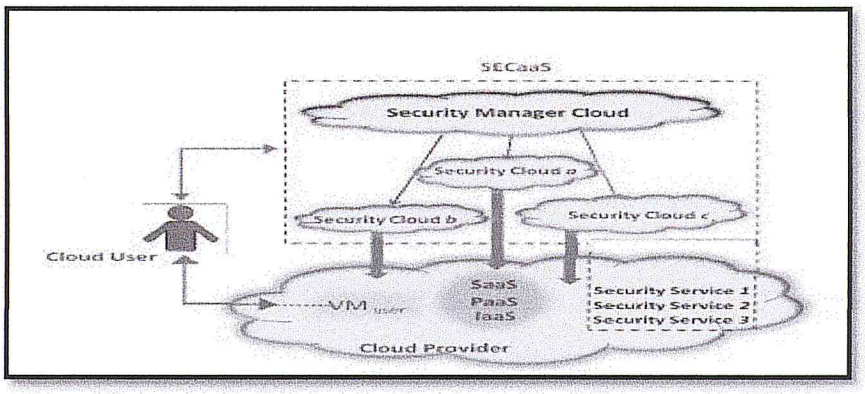


Figure 8 : Service de Cloud Computing *[12]*.

La Figure-8 décrit l'architecture de SecaaS, cette architecture cible la perspective de l'utilisateur, qui sécurise les données et les programmes utilisateur, ainsi que la perspective du fournisseur. Le composant central de l'architecture SecaaS est le gestionnaire de sécurité. Le Cloud du responsable de la sécurité permet aux utilisateurs de choisir les mesures de sécurité nécessaires. Il fixe entre le groupe et les autres Cloud qui offrent des services de sécurité. Il est responsable de la gestion des Cloud de sécurité.

I.7.2- Catégorie de SecaaS

Il existe un grand nombre de produits et de services qui relèvent la sécurité en tant que Service tels que :

- Protection contre le déni de service distribué
- Gestion de la sécurité
- Intrusion Detection/Prevention (IDS/IPS)
- Pare-feu d'applications Web
- Sécurité du courrier électronique (Email Security)
- Évaluation de la sécurité
- Information sur la sécurité et gestion des événements (SIEM)
- Continuité des activités et reprise après sinistre
- Cryptage et gestion des clés

Les services de cryptographie, cryptent les données et / ou gèrent les clés de cryptage. Ils peuvent être offerts par les services Cloud pour prendre en charge le chiffrement et la sécurité des données gérés par le client. Ils peuvent être limités à seulement protéger des actifs au sein de ce fournisseur de Cloud spécifique, ou ils peuvent être accessibles à travers plusieurs fournisseurs (*et même sur site, via API*) pour une gestion plus large du cryptage. La catégorie comprend également chiffrement proxy pour SaaS, qui interceptent le trafic SaaS pour crypter les données discrètes. Cependant, le chiffrement des données en dehors d'une plate-forme SaaS peut affecter la capacité de la plate-forme à utiliser le données en question, dans notre travail nous somme intéressé à la gestion de clés, particulièrement systèmes de gestion de clés cryptographiques CKMS qui nous allons détailler dans le 2eme chapitre [13]

8-Conclusion

Dans ce chapitre, nous avons présenté le Cloud Computing, type de services Cloud, et la sécurité en tant que service (*SecaaS*). Dans notre travail on s'intéresse à un type de service particulier du SecaaS à savoir la gestion des clés de chiffrement/signature à travers un système de gestion de clés de chiffrement (*CKMS*). Dans le chapitre suivant nous allons présenter le CKMS.

CHAPITRE II :
Gestion des clés Cryptographique

Chapitre II: Gestion des clés cryptographique

II.1-Introduction

La sécurité a toujours été un facteur essentiel et crucial dans tous les domaines. La sécurité informatique est l'ensemble des mesures qui permettent de protéger les informations gérées par un système donné, ainsi que les équipements qui permettent de traiter, transmettre et de stocker l'information. Elle doit assurer la disponibilité, l'intégrité, la confidentialité des données. Elle permet aussi d'éviter les indisponibilités, incidents, erreurs, négligences et malveillances. Dans ce chapitre nous allons présenter un background sur les bases de la cryptographie ainsi que les systèmes de gestion des clés de chiffrement.

II.2 –Bases de la cryptographie

Dans cette partie nous allons présenter un background sur les bases cryptographique.

II.2.1- Propriétés de la sécurité

- **Confidentialité** : Les données, l'objet ou les acteurs de la communication ne peuvent pas être connues d'un tiers non-autorisé (*utilisation d'un algorithme de chiffrement*).
- **Authenticité** : L'identité des acteurs de la communication est vérifiée (*utilisation d'algorithmes d'authentification*).
- **Intégrité** : Les données de la communication n'ont pas été altérées, modifiées, ou détruites tant de façon intentionnelle qu'accidentelle.
- **Non-répudiation** : Les acteurs impliqués dans la communication ne peuvent nier y avoir participé (*utilisation d'algorithmes de signatures*).
- **Disponibilité** : Les acteurs de la communication accèdent aux données dans de bonnes conditions [14].

II.2.2 -Vocabulaire de base

- **Cryptologie :** Il s'agit d'une science mathématique comportant deux branches : la cryptographie et la cryptanalyse [15].
- **Cryptographie :** La cryptographie est l'étude des méthodes donnant la possibilité d'envoyer des données de manière confidentielle sur un support donné [15].
- **Chiffrement :** Le chiffrement consiste à transformer une donnée (texte, message, ...) afin de la rendre incompréhensible par une personne autre que celui qui a créé le message et celui qui en est le destinataire. La fonction permettant de retrouver le texte clair à partir du texte chiffré porte le nom de déchiffrement [15].

- **Texte chiffré :** Appelé également cryptogramme, le texte chiffré est le résultat de l'application d'un chiffrement à un texte clair [15].
- **Clé :** Il s'agit du paramètre impliqué et autorisant des opérations de chiffrement et/ou déchiffrement [15].
- **Cryptanalyse :** Opposée à la cryptographie, elle a pour but de retrouver le texte clair à partir du texte chiffré en déterminant les failles des algorithmes utilisés [15].
- **Crypto-système :** Il est défini comme l'ensemble des clés possibles (espace de clés), des textes clairs et chiffrés possibles associés à un algorithme donné [15].
- **Module cryptographique :** un ensemble d'équipements et de logiciels qui implémentent des fonctions de sécurité (incluant les algorithmes de chiffrement et la génération des clés) [16].
- **Random Bit Generator (RBG) :** Outil ou algorithme qui donne en sortie une série de bits aléatoires [16].

Notations : La propriété de base est que M = D(E(M)), où :

- M : le texte en clair
- C : le texte chiffré
- K : la clé (dans le cas d'un algorithme à clé symétrique), Ek et Dk dans le cas d'algorithmes asymétriques
- E(x) : la fonction de chiffrement, et
- D(x) : la fonction de déchiffrement.

Ainsi, avec un algorithme à clef symétrique M = D(C) si C = E(M) [15] .

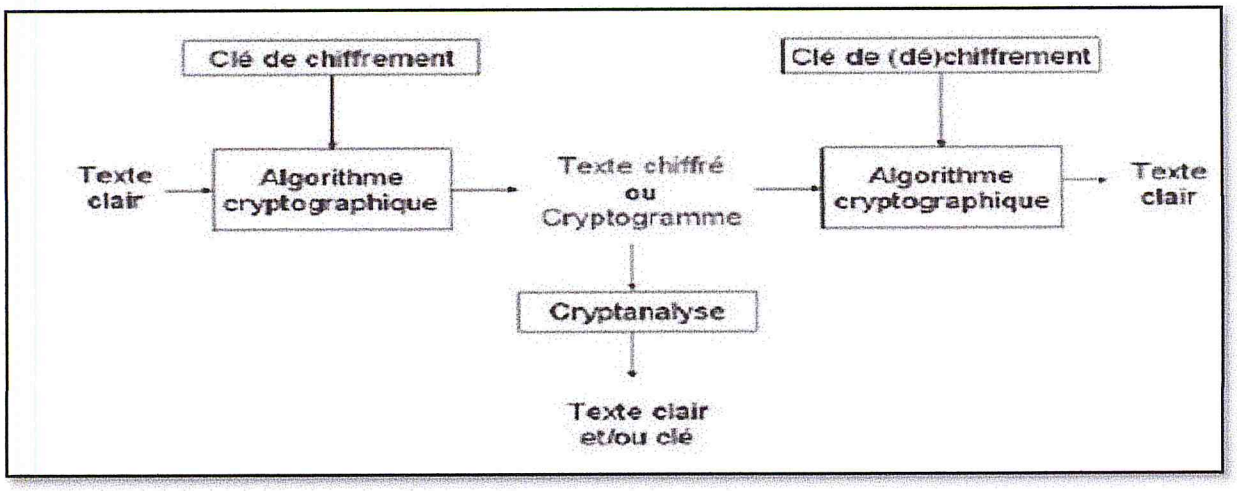


Figure 9 : Protocole de chiffrement[15]

II.2.3 -Types de menaces et attaques

Il existe principalement deux types de menaces : [17]

- Menace passive : menace la confidentialité. Une information sensible parvient également à une personne autre que son destinataire légitime.
- Menace active : menace l'intégrité de l'information, le message échangé peut être modifié.

On distingue quatre types d'attaque : [17]

- **Texte chiffré connu** : seul C est connu de l'attaquant
- **Texte clair connu** : l'attaquant connait C et M correspondant
- **Texte clair choisi** : ∀ M, l'attaquant peut obtenir C
- **Texte chiffré choisi** : ∀ C, l'attaquant peut obtenir M.

II.2.4- Algorithmes de chiffrement

Il existe deux grandes familles d'algorithmes cryptographiques à base de clés : les algorithmes à clés secrète ou algorithmes symétriques, et les algorithmes à clés publique ou algorithmes asymétriques.

II.2.4.1 -Chiffrement symétrique

La clé de chiffrement est la même que la clef de déchiffrement, et elle ne doit être connue que des tiers communicants et d'eux seul, elle est dite **la clé secrète** [18].

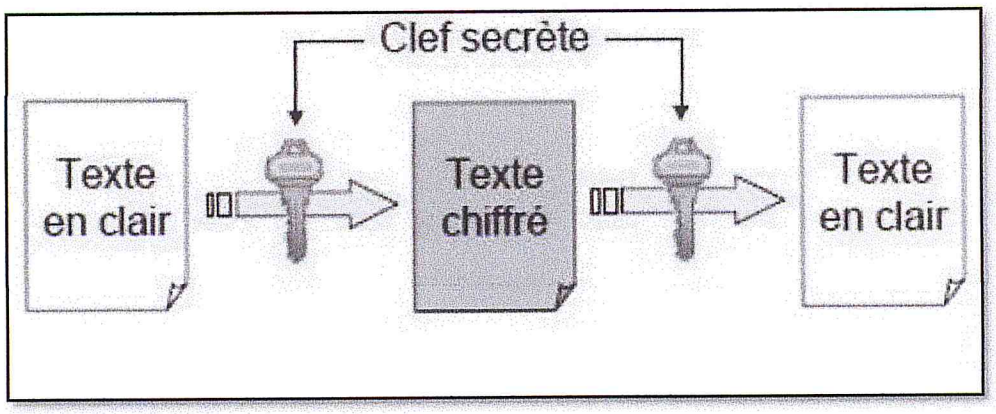


Figure 10 : Chiffrement symétrique[18]

Il existe plusieurs algorithmes de type symétrique : Algorithmes de chiffrement en continu ou par flux (stream cipher), par blocs (block cipher), opèrent sur le texte en clair par blocs (généralement, 64 bits).

Exemple : DES, 3DES, AES [18].

II.2.4.2- Chiffrement à clef publique

Avec les algorithmes asymétrique, les clésde chiffrement et de déchiffrement sont distinctes et ne peuvent se déduire l'une de l'autre. On peut donc rendre l'une des deux publique tandis que l'autre reste privée. C'est pourquoi on parle de chiffrement à clés publique. Si la clef publique sert au chiffrement, tout le monde peut chiffrer un message, que seul le propriétaire de la cléprivée pourra déchiffrer. On assure ainsi la confidentialité. Certains algorithmes permettant d'utiliser la cléprivée pour chiffrer. Dans ce cas, n'importe qui pourra déchiffrer, mais seul le possesseur de la clef privée peut chiffrer. Cela permet donc la signature de message [18].

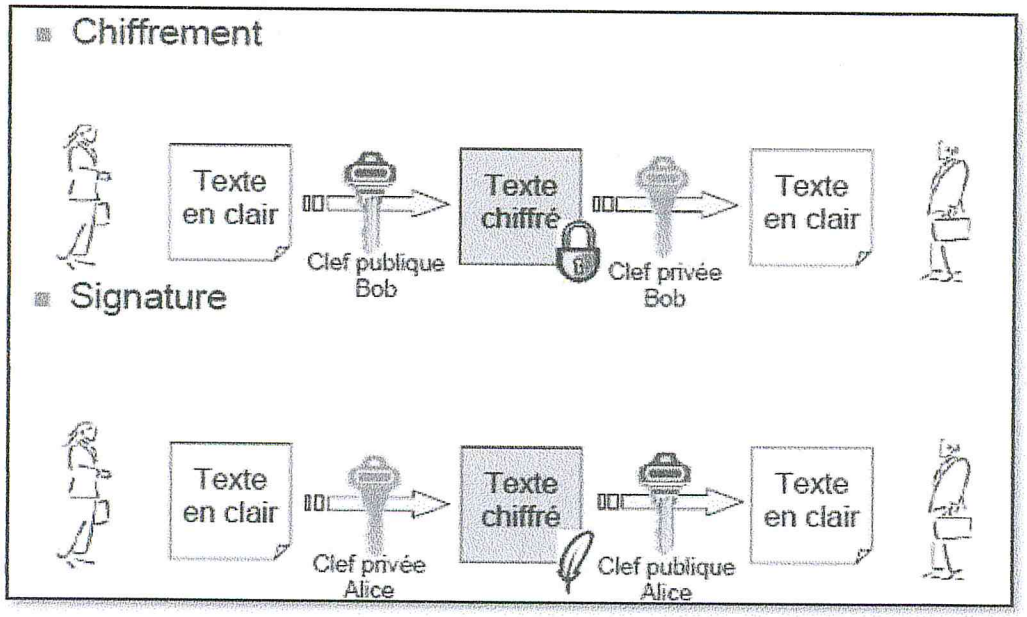


Figure 11 : Cryptographie à clé publique[18]

Les principaux algorithmes sont : Merkle-Hellman-Diffie 1976, RSA : RivestShamirAdleman 1978, El Gamal[18]

II.2.4.3-Clés de chiffrement

La clé de chiffrement devrait être générée au niveau d'un module cryptographique, si une suite de bits aléatoires est requise ; on utilise le RBG qui doit être implémenté au niveau du module. Une fois la clégénérée, elle doit être transmise utilisant des canaux sécurisés et utilisé par l'algorithme associé [16].La force de sécurité supportée par une clé dépend de plusieurs paramètres :L'algorithme, la taille de la clé, le processus de génération de la clé et comment la clé a été prise en charge[16].

La plupart des algorithmes de cryptographie utilisent des clés : c'est le cas des algorithmes de chiffrement symétrique (AES) ou asymétrique (RSA). Un paramètre important

implémentation est le choix de la taille de ces clés. Ce choix affecte tout d'abord la sécurité. Plus une clé est longue, plus les attaques contre l'algorithme seront complexes.

Dans le cas des algorithmes de chiffrement symétrique, la taille des clés à choisir est directement liée à la complexité d'une recherche exhaustive. Par exemple, Pour retrouver une clef AES de 128 bits, vous aurez besoin d'essayer 2127 clés en moyenne (*il y a 2128 clés possibles et en moyenne vous trouverez la bonne au bout de la moitié de vos essais*).

Dans le cas des algorithmes de chiffrement asymétrique et des algorithmes de signature digitale, la taille des clés à choisir dépend des propriétés mathématiques de l'algorithme et il existe de bien meilleures attaques que la recherche exhaustive. Par exemple, dans le cas de RSA, factoriser le module n de la clé publique permet de retrouver la clé privée. A l'heure actuelle, la taille minimum acceptable pour de telles clés est 1024 bits et la taille recommandée est de 2048 bits [19].

II.2.5 Fonction de hachage, signature et scellement

Il existe plusieurs mécanismes qui fournissent des services d'intégrité, d'authentification de l'origine des données et de non répudiation de la source [18] :

II.2.5.1 -Fonction de hachage

Aussi appelée fonction de condensation permet de s'assurer de l'intégrité du message, une fonction de hachage est une fonction qui convertit une chaine de longueur quelconque en une chaine de taille inférieure et généralement fixe ; la chaine résultante est appelée empreinte (*digest en anglais*) ou condensé de la chaine initiale. Elle possède les propriétés suivantes :

- Étant donné m, il est facile de calculer $h(m)$
- Étant donné h, il est difficile de calculer m tel que $h(m)=h$
- Étant donné m, il est difficile de trouver un autre message, m', tel que $h(m)=h(m')$ [18].

Les fonctions de hachage les plus répandues sont :

- MD5 (Message Digest 5) : Empreinte de 128 bits.
- SHA (Secure Hash Algorithm) :
- Norme NIST.
- Empreinte de 160 bits.
- SHA-1 considéré comme plus sûr que MD5(1994).
- SHA-2 (octobre 2000) agrandit la taille de l'empreinte (SHA-256, SHA-512) [18].

II.2.5.2 -Signature numérique

Une signature numérique fournit les services d'authentification de l'origine des données, d'intégrité des données et de non-répudiation.

La méthode utilisée pour signer consiste à calculer une empreinte du message à signer et à ne chiffrer que cette empreinte [18].

Les algorithmes utilisés sont :

- DSS : standard NIST
- SHA-1 + El-Gamal
- RSA :
- MD5 + RSA
- SHA-1 + RSA **55** [17]

II.2.5.3 -Scellement

Tout comme la signature numérique, le scellement fournit les services d'authentification de l'origine des données et d'intégrité des données, mais il ne fournit pas la non-répudiation. Ceci permet l'utilisation de la cryptographie à clef secrète pour la génération du sceau ou code d'authentification de message.

Un code d'authentification de message (*Message Authentification Code, MAC*) est le résultat d'une fonction de hachage à sens unique dépendant d'une clef secrète, l'empreinte dépend à la fois de l'entrée et de la clef. On peut construire un MAC à partir d'un algorithme de chiffrement par blocs ou d'une fonction de hachage (*HMAC*) [18].

II.3 –Système de gestion des clés de cryptage (CKMS)

Dans toutes les industries, les exigences de gestion des clés cryptographiques deviennent de plus en plus complexes. S'assurer que la bonne clé est au bon endroit au bon moment et se conformer à des procédures rigoureuses de gestion de la sécurité est mandaté par de nombreuses organisations. Ces exigences peuvent être très difficiles, car la plupart des entreprises ont besoin de gérer un nombre toujours croissant de clés, tout en réduisant le risque de fraude interne et externe, tout en limitant les coûts au minimum.

Le système de gestion des clés de cryptage (CKMS) rationalise l'administration et réduit les coûts associés à la gestion des clés traditionnelle basée sur le papier et les XOR. Grâce à ses protocoles efficaces et automatisés, CKMS offre aux utilisateurs la flexibilité de gérer un très grand nombre de clés d'application - tout au long de leur cycle de vie - sans se noyer de travail. [20]

Dans cette section on va présenter CKMS, son concept, ses fonctions de gestion des clés ainsi que sa sécurité.

II.3.1-Définition de CKMS :

Un Système de gestion des clés de chiffrement ou CKMS (*Cryptographic Key Management System*) est une plateforme de gestion de clés de chiffrement qui se compose de plusieurs composants et dispositifs hardware et software qui fonctionnent selon des politiques et procédures, utilisés pour contrôler la gestion des clés de chiffrement et certaines informations spécifiques qui leurs sont associés. Un CKMS peut être construit pour servir une application particulière (par exemple, E-mail, stockage de données, les systèmes de santé et les systèmes de paiement), ou il peut être conçu pour servir toute une entreprise, qui englobe plusieurs applications.

Lors de la conception d'un CKMS, les techniques cryptographiques utilisées pour protéger les clés gérées, devrait offrir un niveau de protection appelé le niveau de sécurité qui devrait-être impossible pour un attaquant de contourner ou subvertir [16] .

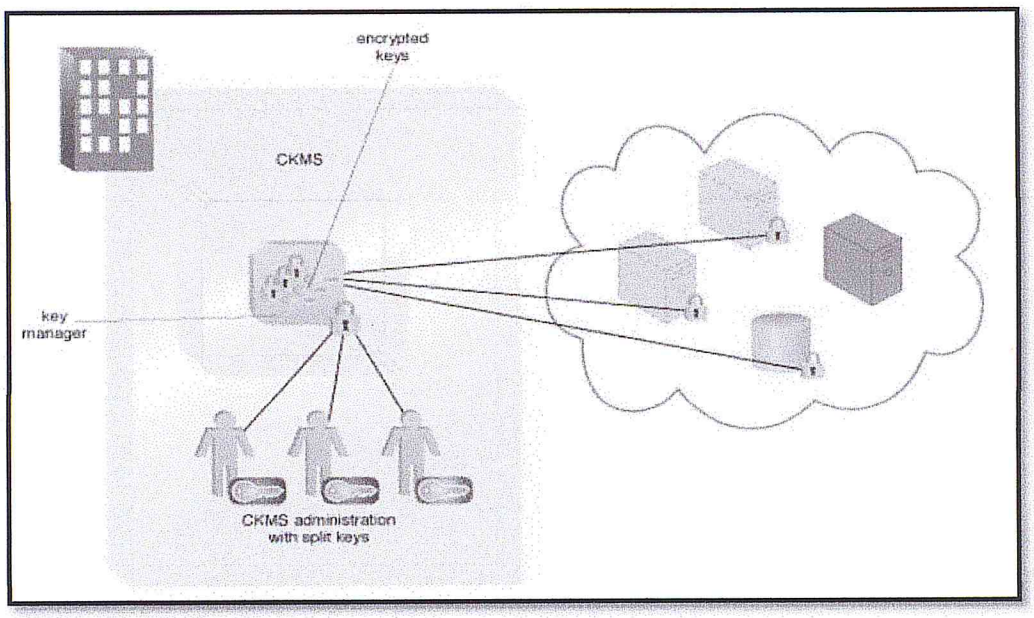


Figure 12 : Exemple de CKMS utilisant l'administration de clé partagée. *[21]*

La Figure-12 montre un CKMS gérant les clés de l'organisation. De nombreux règlements de l'industrie exigent que les organisations contrôlent leurs propres clés, ce qui peut être fait sur place ou à partir d'un service tiers de confiance. Il montre la nécessité pour plusieurs administrateurs d'accomplir des tâches de gestion nécessitant plusieurs clés fractionnées pour effectuer une opération cryptographique.

II.3.2 - Architecture d'un CKMS

L'architecture d'un CKMS peut être simple ou complexe, un logiciel fonctionnant sur un ordinateur pour utilisateur, ou une variété de sous-systèmes contenant chacun de nombreux dispositifs (matériels, logiciel, microprogrammes...) qui fournissent des services de gestion de clés pour de nombreux utilisateurs du réseau et des applications largement distribuées géographiquement et reliées à une myriade de réseaux de communication. Un CKMS peut offrir des services de gestion tiers en travaillant en parallèle avec des entités telles qu'une autorité de certification (CA), un centre de distribution des clés (KDC) ainsi qu'une autorité d'enregistrement (RA) [16].

Le design d'un CKMS devrait répondre à certaines conditions :

- Assurer une gestion des clés, des applications et des utilisateurs.
- Conformité aux normes (pour augmenter la confiance en la conception du CKMS).
- Facilité d'utilisation (le CKMS doit être aussi transparent que possible, avec des interfaces utilisateur simple à utiliser).
- Performances et évolutivité (le CKMS doit être évolutif afin de répondre à la croissante charge de travail) [16].

La figure 13 représente un schéma d'une architecture simplifiée d'un CKMS :

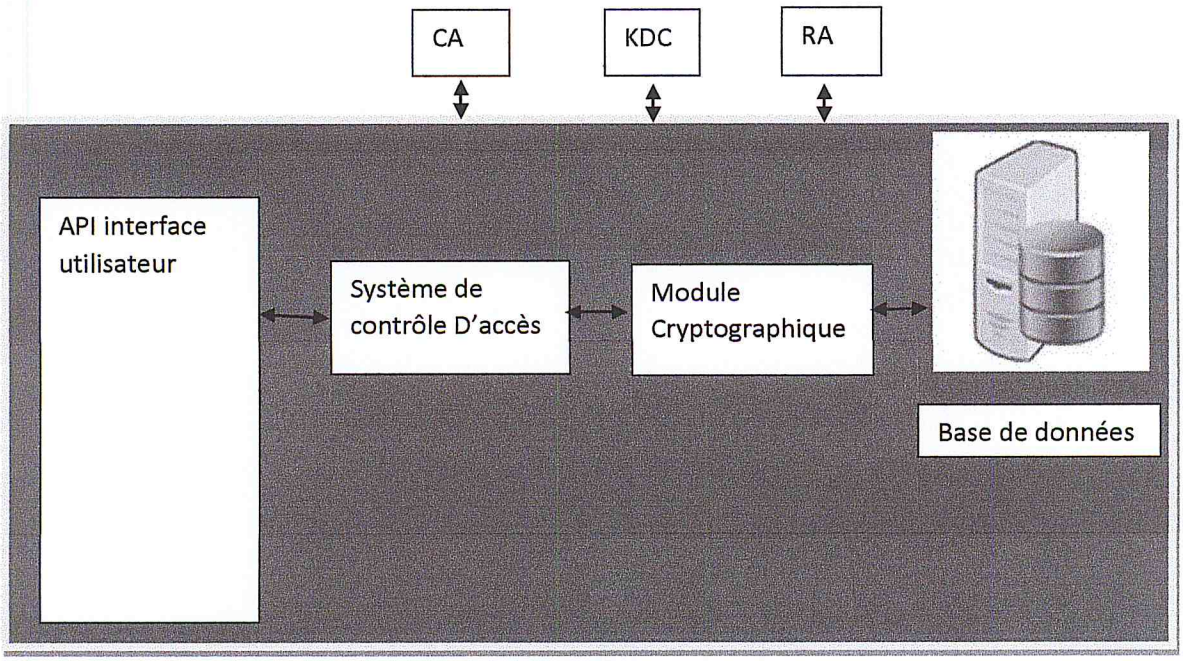


Figure 13 : Architecture simplifiée d'un CKMS

II.3.3 - Rôles et responsabilités

Un CKMS peut avoir une interface avec les humains qui occupent certains postes de gestion ou des rôles opérationnels. Chaque rôle procure un accès à un ensemble de fonctions de gestion des clés qui sont nécessaires pour s'acquitter des responsabilités du rôle.

Les rôles possibles du CKMS sont les suivants:

- **Autorité système :** Responsable de la gestion des postes de direction pour l'ensemble des opérations et de la sécurité d'un CKMS. Une autorité système gère tous les rôles opérationnels CKMS. Un rôle opérationnel est un rôle qui exploite directement le CKMS.
- **Administrateur système:** Responsable du personnel, le fonctionnement quotidien, la formation, la maintenance et la gestion d'un CKMS autre que ses clefs (*établir l'identité de toutes les personnes impliquées dans le fonctionnement et l'utilisation d'un CKMS*).
- **Agent cryptographique :** Un agent cryptographique est autorisé à effectuer la fonction d'initialisation de chiffrement et de gestion sur le module cryptographique.
- **Une autorité de domaine :** Chargée de déterminer les conditions nécessaires pour la communication avec un autre domaine de sécurité.
- **Propriétaire de la clé :** c'est une entité (*par exemple, une personne, un groupe, une organisation, un dispositif ou module*) autorisé à utiliser une clé ou une paire de clefs et dont l'identifiant est associé à une clé ou une paire de clés.
- **Les utilisateurs système:** Les utilisateurs du système (*souvent les propriétaires des clés*) utilisent les CKMS lorsque les fonctions de gestion des clés sont nécessaires pour une application.
- **Administrateur d'audit :** un administrateur de vérification est chargé de vérifier tous les aspects d'un CKMS (*sécurité et les opérations autorisées*).
- **Agent d'enregistrement :** Un agent d'enregistrement est responsable de l'enregistrement des nouvelles entités et de l'association des clés à leurs identifiants.
- **Agent de récupération de clef :** un agent de récupération de clé est autorisé à récupérer les clés de stockage de sauvegarde ou d'archivage après vérification de l'identité et de l'autorisation de l'entité requérante conformément à la politique de sécurité CKMS.
- **Opérateur CKMS :** un opérateur CKMS est autorisé à exploiter un CKMS à la place de l'administrateur système.
- Plusieurs personnes peuvent être affectées à chaque rôle, et une seule personne ne peut jouer plusieurs rôles à la fois [16].

III.3.4- Dictionnaire de données du CKMS (D-CKMS)

Le Dictionnaire de données est défini comme l'ensemble des clés et leurs informations associées qui sont explicitement enregistrées et gérées par le CKMS. Le Dictionnaire de données est considéré comme le noyau du CKMS.

Chaque clé possède des informations qui lui sont associées, qui spécifient les caractéristiques, des utilisations possibles et les paramètres applicables sur la clé. Spécifiant par exemple le type de la clé, comment elle a été générée, quand est ce qu'elle a été générée, son identifiant, l'algorithme pour lequel elle a été générée et son crypto-période. Tout comme les clés, leurs informations associées doivent être protégées contre les modifications non-autorisées et avoir une source bien authentifiable [16] .

En général, les clés cryptographiques sont classées selon leurs propriétés et utilisations. Les clés peuvent être publiques, privées ou symétriques, elles peuvent aussi avoir des propriétés statiques ou bien à effet éphémère (*utilisé seulement pour une seule session ou bien une seule transaction*).

L'utilisation de la clé inclus : la signature, l'authentification, le chiffrement / déchiffrement, la couverture de la clé, *RNG* (*Random Number Generation*), la clé maîtresse, la clé de transport [16].

Le dictionnaire de données est constitué des éléments suivants [16]:

- Label de la clé
- Identificateur du propriétaire
- Spécificateur du format de la clé
- Algorithme de chiffrement utilisant la clé
- Taille de la clé
- Politiques de sécurité applicables au type de clé
- La clé père
- Dates-Temps
- Identificateur de la clé
- L'état de cycle de vie d'une clé
- Produit utilisé pour créer la clé
- Paramètres de la clé
- Type de clé
- Liste de Control d'Accès des clés (ACL)
- protection des clés

II.3.5- Fonctions de gestion du D-CKMS

Plusieurs fonctions sont exécutées par le CKMS pour des fins de gestion des clés (ou bien les informations qui leurs sont associées), l'authentification de l'entité qui souhaite exécuter une

fonction et l'autorisation affirmative ou négative d'exécution est effectuée par un système de controle d'accès (ACS), parmi les fonctions on peut citer [16]:

- Stockage de la clef opérationnel et des informations associées.
- Affichage de l'information associée d'une clé.
- Récupération des clefs et/ou ses informations associées.
- Génération d'une clé.
- Enregistrement du propriétaire.
- Insertion d'une clé et de ses informations associées dans un module cryptographique.
- Désactivation d'une clé.
- Mise à jour d'une clé.
- Destruction d'une clé et de ses informations associées.
- Modification de l'information associée à une clé.
- Exécution d'une fonction cryptographique utilisant une clé (génération d'une signature numérique, chiffrement, déchiffrement...)

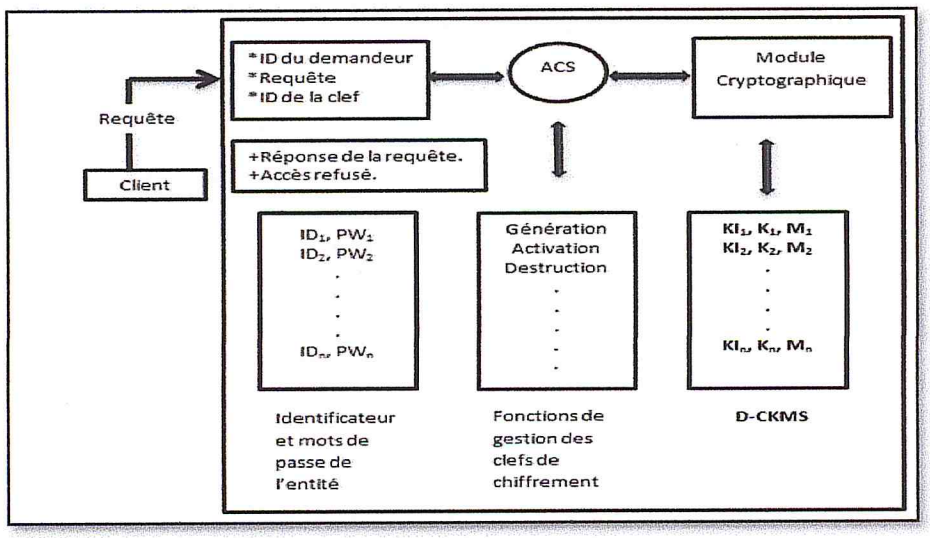


Figure 14: Processus d'opération d'une fonction de gestion de clef de chiffrement *[16]*

La Figure-14 illustre l'opération d'une fonction de gestion de clef de chiffrement. Faire appel à une fonction se traduit par l'envoie de l'identificateur de l'entité (client), le nom de la fonction, et l'identificateur de la clé, qui sont ensuite envoyés au système de controle d'accès (ACS), après authentification de l'entité, une vérification de l'autorisation d'exécution de la fonction est effectuée, en vérifiant que l'ID de l'entité se trouve bien au niveau de la liste de contrôle d'accès (ACL se trouvant au niveau du D-CKMS) correspondant à la clé et à la fonction. Si l'ACS trouve que la requête émise par l'entité doit être refusée, il lui renvoie un

avis défavorable, sinon la requête est acceptée, la fonction sera exécutée et la réponse de la fonction sera retournée au client [16].

II.3.6- Politiques de sécurité du CKMS

Une politique de sécurité CKMS est créée pour établir et préciser les exigences de protection de la confidentialité, l'intégrité, la disponibilité et l'authentification source au cours du cycle de vie de toutes les clefs utilisées par l'organisation. Elle comprend la sélection de tous les mécanismes et protocoles cryptographiques pouvant être utilisés dans les systèmes de l'organisme d'information automatisés. Les politiques de sécurité CKMS peuvent être spécifiées dans un formulaire (par exemple, des tableaux, diagrammes,..) de sorte qu'un CKMS peut automatiquement appliquer tout ou une partie de la politique.

La figure 15 présente les points forts de la sécurité de CKMS qui sont la connexion entre les différentes politiques de la sécurité de l'environnement.

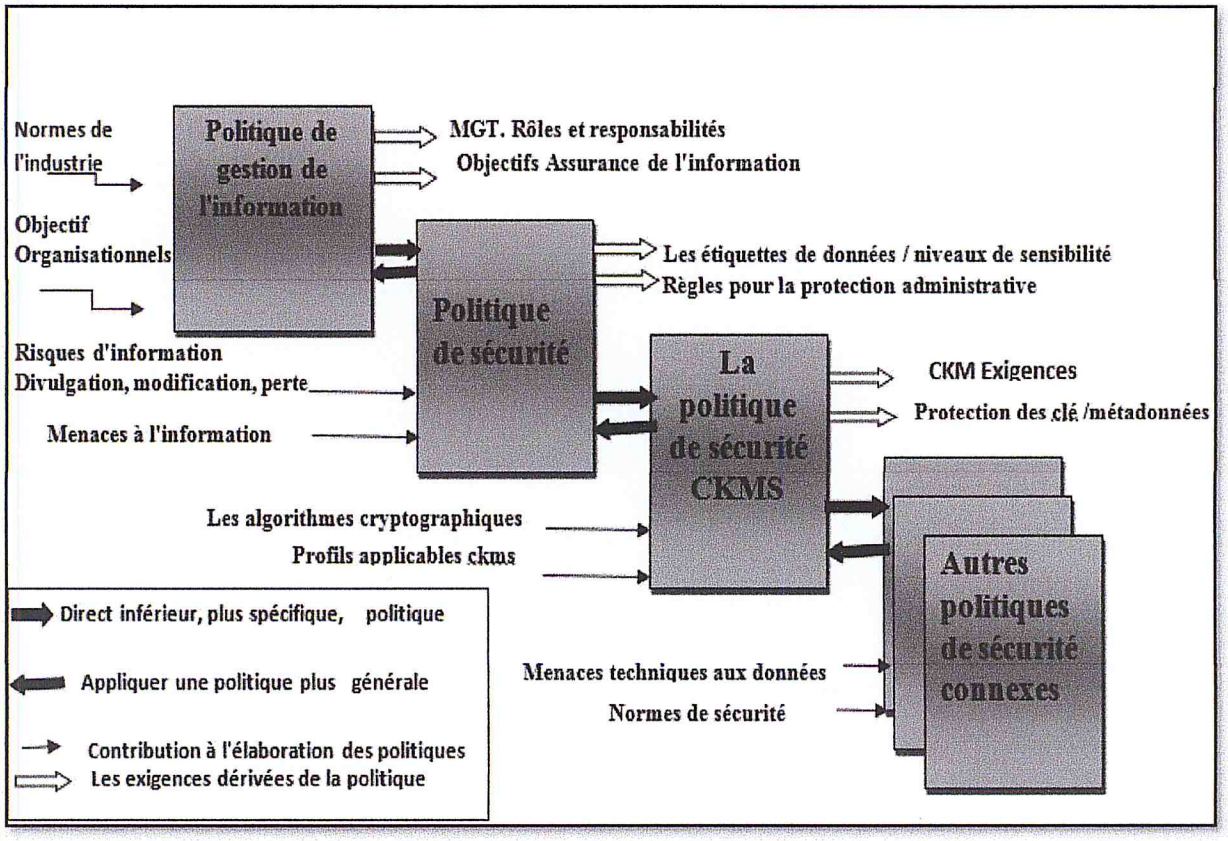


Figure 15: Politiques de sécurité connexes [16]

II.4 -Conclusion

Dans ce chapitre nous avons vu les différentes propriétés de la sécurité et les différents algorithmes et techniques utilisés pour garantir la sécurité des données, par ce fait nous nous sommes familiarisés avec un vocabulaire qui nous sera utile dans la suite de ce travail.

La sécurité d'un système de chiffrement repose sur la sécurisation des clefs de chiffrement gérée par un système dédié (*plateforme*), qui assure leurs gestions tout au long de leurs cycles de vie. Cette plateforme appelée un système de gestion de clefs de chiffrement (CKMS). Ce système confronté à plusieurs enjeux de sécurité surtout dans un contexte virtuel (*Cloud-Computing*). A cette effet et pour plus de confidentialité et la sécurité des clés de chiffrement, on doit renforcer la sécurité grâce à l'utilisation d'un équipement physique appelé Trusted Platform Module (*TPM*) qu'on va donc présenter dans le chapitre suivant

CHAPITRE III: Module de Plateforme de Confiance

Chapitre III: Module de Plateforme de confiance (TPM)

III.1-Introduction

Le Module de plateforme de confiance (*Trusted Platform Module TPM*), est un composant hardware présent sur la plupart des PC et des serveurs. Ce composant utilisé pour la gestion des intégrités et l'authenticité des applications qui s'exécutent sur une machine physique (*PC ou serveurs*).

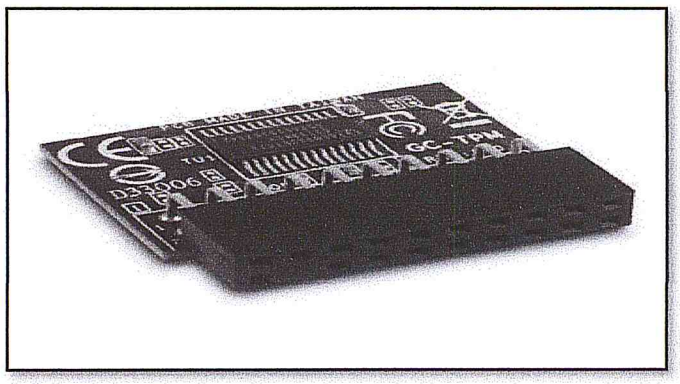


Figure 16 : Trusted Platform Module (*TPM 2.0*) *[22]*

Le TPM est conçu pour fournir des fonctions liées à la sécurité matérielle. Une puce TPM est un crypto-processeur sécurisé qui aide avec des actions telles que la génération, le stockage et la limitation de l'utilisation de clés cryptographiques. De nombreux TPM incluent plusieurs mécanismes de sécurité physique pour le rendre inviolable, et les logiciels malveillants ne peuvent pas altérer les fonctions de sécurité du TPM.

Traditionnellement, les TPM étaient des puces discrètes soudées à la carte mère d'un ordinateur. De telles mises en œuvre permettent, en tant que fabricant d'équipement d'origine, d'évaluer et de certifier le TPM distinct du reste du système. Certaines implémentations TPM plus récentes intègrent la fonctionnalité TPM dans le même jeu de composants que d'autres composants de plate-forme tout en offrant une séparation logique similaire aux puces TPM discrètes.

Les TPM sont passifs : ils reçoivent des commandes et retournent des réponses. Pour tirer pleinement parti d'un TPM, on doit soigneusement intégrer le matériel et le microprogramme du système au TPM pour lui envoyer des commandes et réagir à ses réponses. Les modules

TPM offrent des avantages en termes de sécurité et de confidentialité pour le matériel système, les propriétaires de plate-forme et les utilisateurs [23].

Dans ce chapitre nous allons présenter l'architecture du TPM, ses fonctionnalités ainsi que ses avantages pour la partie cryptographie, clés de chiffrement et algorithmes et enfin son utilisation dans le Cloud-Computing.

III.2-Historique de TPM

Le premier TPM largement déployé était TPM 1.1b, qui a été publié en 2003. À cette date au début, les fonctions de base d'un TPM qui étaient disponibles sont : la génération de clés inclus, le stockage, l'autorisation sécurisée et attestation dispositif santé. Un inconvénient de TPM 1.1b est-elle contient des incompatibilités au niveau du matériel. Les fournisseurs de TPM ont des interfaces légèrement différentes, ce qui nécessite des pilotes différents et un brochage du paquet qui ne sont pas standardisés

TPM 1.2 a été développé dans la période 2005-2009, il a traversé plusieurs versions. Ses améliorations initiales sur 1.1b inclus une interface logicielle standard et un Brochage paquet la plupart du temps standard (TCG). La spécification TPM 1.2 exige que les TPM ont une protection contre les attaques par dictionnaire, et groupes de confidentialité se sont plaints de l'absence de mise en œuvre de CA de la vie privée. Cela a conduit à l'inclusion dans TPM 1.2 de nouvelles commandes pour une deuxième méthode de touches anonymisation pour aider à répondre à cette préoccupation. Il est avéré que l'envoi d'une machine avec un certificat utilisait le TPM, par exemple l'approbation sur le disque dur était dans de nombreux cas, peu pratique, parce que les organisations souvent effacées du disque dur quand ils ont reçu et installé leur propre charge logicielle. Quand ils l'ont fait, le certificat a été supprimé. Afin de fournir une solution, une petite quantité de RAM non volatile (habituellement environ 2 Ko) a été ajouté dans TPM 1.2; il avait des contrôles d'accès spécialisés ainsi que d'un petit nombre de compteurs monotones.

En 2014, après les faiblesses cryptographiques de SHA-1 ont causé la nécessité d'un changement vers le **TPM 2.0**, l'architecture a été remanié de éraflure résultant dans une conception beaucoup plus intégrée et unifiée [24].

III.3-Architecture de TPM :

On peut résumer l'architecture de TPM dans la figure suivante :

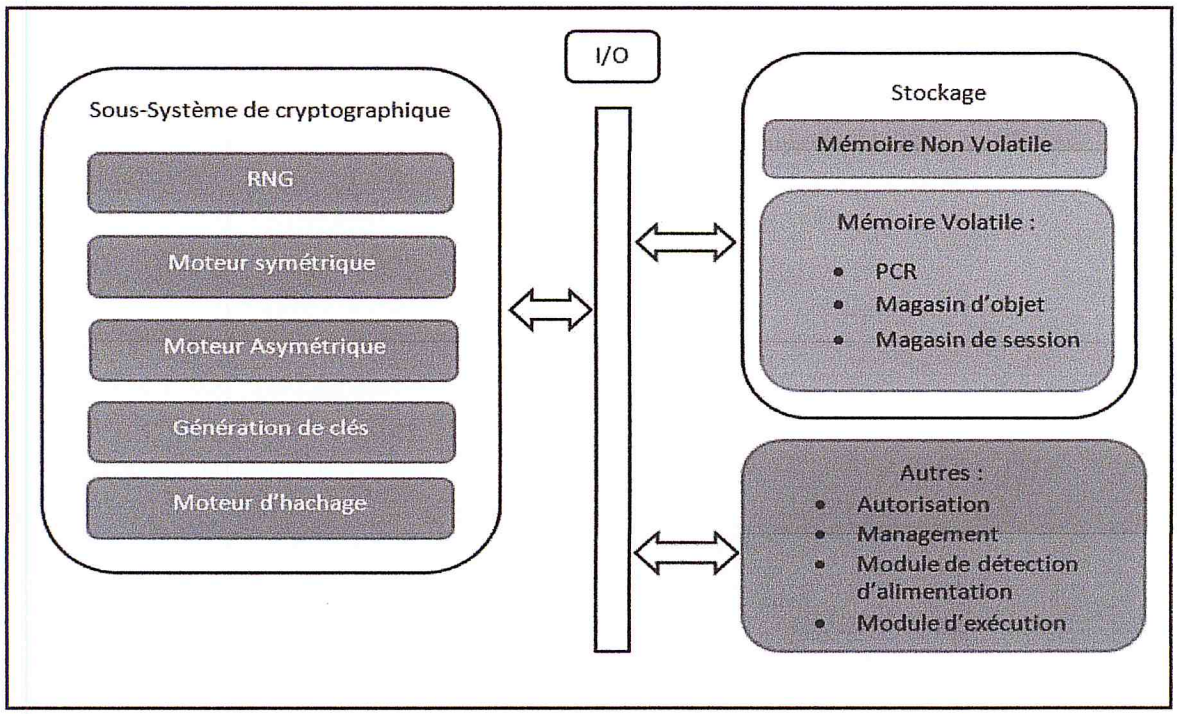


Figure 17 : Architecture de TPM 2.0 [23]

III.3.1-Sous-système cryptographique de TPM

Le sous-système de cryptographie implémente les fonctions cryptographiques du TPM. Il peut être appelé par le module de commande de l'analyse, l'autorisation du Sous-système ou l'exécution de la commande module. Le TPM utilise des opérations cryptographiques classiques dans des moyens classiques. Ces opérations comprennent

- La génération de clés : RNG, Symétrique, Asymétrique.
- Les fonctions de hachage (*Hash Message Authentication Code HMAC*).
- Le chiffrement et le déchiffrement (RSA, AES, 3DS…)
- Signature et vérification de signature.

III.3.2-Stockage dans le TPM

- **Mémoire non volatile (NVRAM)** : Le module de mémoire NV stocke l'état persistant associé au TPM. Certaines mémoires NV peuvent être allouées et utilisées par la plate-forme et les entités autorisées par le propriétaire du TPM. Parmi les éléments de NVRAM:
  - Stockage des clés de racine pour les chaînes de certificats

- Stockage pour une clé d'approbation (EK)

- **Mémoire à accès aléatoire (Volatile Memory)** : La mémoire vive (RAM) contient les données transitoires TPM. Les données contenues dans la RAM TPM sont autorisées, mais non requises à être perdues lorsque l'alimentation TPM est supprimée. Étant donné que les valeurs contenues dans la RAM TPM peuvent être perdues, elles sont qualifiées de volatiles dans cette spécification, même si la perte de données dépend de l'implémentation. [23]

III.3.3-Autres

- **I/O Buffer** : C'est le domaine de la communication entre un TPM et le système hôte. Le système de données de commande dans le tampon d'E/S et récupère les données de réponse de la mémoire tampon. Il n'est pas nécessaire que le tampon d'E/S soit physiquement isolé des autres parties du système. Il peut être une mémoire partagée. Cependant, lors du traitement d'une commande, la mise en œuvre doit s'assurer que le TPM est en utilisant les valeurs correctes. [23]
- **Module de détection d'alimentation** : Ce module gère les états d'alimentation TPM en conjonction avec les états d'alimentation de la plate-forme de telle sorte le TPM doit être informé de tous les changements d'état d'alimentation.
- **Sous-système d'autorisation** : Le sous-système d'autorisation est appelé par le module Command Dispatch au début et à la fin de l'exécution de la commande. Avant que la commande puisse être exécutée, le sous-système d'autorisation vérifie que l'autorisation appropriée pour l'utilisation de chacun des emplacements protégés a été fournie. [23]

III.4-Algorithmes de TPM

À partir de TPM 2.0, la spécification permet une grande flexibilité dans les algorithmes cryptographique que TPM peut utiliser. Un module TPM peut utiliser pratiquement tout algorithme de hachage, les algorithmes symétriques comme AES, et de nouveaux algorithmes asymétriques tels ECC, RSA. Le TPM 2.0 permet tout type d'algorithme de chiffrement. Cela signifie que si un autre algorithme est affaibli par la cryptanalyse à l'avenir, la spécification ne sera pas nécessaire de changer, Les algorithmes clés devraient idéalement correspondre à la force. Table 1 énumère les principaux atouts d'algorithmes. [24]

Type	Algorithme	Bits
Asymétrique	RSA 1024	80
Asymétrique	RSA 2048	112
Asymétrique	RSA 3072	128
Asymétrique	RSA 16384	256
Asymétrique	ECC 224	112
Asymétrique	ECC 256	128
Asymétrique	ECC 384	192
Asymétrique	ECC 521	260
Symétrique	DES	56
Symétrique	3DES (2 clés)	127
Symétrique	3DES (touche 3)	128
Symétrique	AES 128	128
Symétrique	AES 256	256
Hachage	SHA-1	65
Hachage	SHA 224	112
Hachage	SHA 256	128
Hachage	SHA 384	192
Hachage	SHA 512	256
Hachage	SHA-3	Variable

Tableau 1 : Les Algorithmes d'agilité de TPM 2.0 [24]

III.5-TPM dans le Cloud Computing

Le problème de la sécurité du Cloud Computing devient de plus en plus difficile, l'informatique de confiance est devenue une technologie importante pour améliorer la sécurité de l'environnement de virtualisation de l'infrastructure Cloud.

En effet, le TPM peut être utilisé en tant que moyen ou plateforme de confiance pour la gestion de l'intégrité et l'authenticité des softwares exécutables dans le Cloud-Computing. Cependant, le TPM est conçue pour une seule machine physique avec un système d'exploitation unique, à cet effet, plusieurs techniques de virtualisation proposées pour la création des virtuelle

TPM (instances), et cela afin que soit chaque instance (***vTPM***) soit utilisée par une seule machine virtuelle. [25]

III.6-Conclusion

Dans ce chapitre on a présenté TPM, qui est un composant hardware qui offre plusieurs fonctionnalités de sécurité telle que : la gestion d'intégrité des softwares (*problème d'authenticité*), et un bouquet d'outil cryptographique pour le chiffrement/déchiffrement, signature et la création des clés. En effet le TPM est considéré comme source de confiance et sûre pour la création des clés de chiffrement/signature. Cette capabilité est très intéressante pour un système CKMS délivré en tant que SecaaS dans le Cloud-Computing. Dans le prochain chapitre en va présenter une approche d'un CKMS en tant que SecaaS basé sur le TPM.

CHAPITRE IV : CKMS en tant que SecaaS

Chapitre IV: CKMS en tant que SecaaS

IV.1 -Introduction

Notre travail consiste à la conception et le développement d'une solution de gestion de clés de chiffrement en tant que service de sécurité dans le Cloud Computing. La solution est basée sut le TPM en tant que base de confiance pour la création des clés de chiffrement & signature. Le TPM offre la confidentialité et l'authenticité des clés de chiffrement. A cet effet, dans ce chapitre nous allons présenter la partie de conception, divisé en deux parties :

La 1ere partie est consacrée à l'architecture du CKMS as a SecaaS, ainsi que le processus de création des clés.

La 2ème partie est consacrée sur la modélisation : Dans cette partie nous avons utilisé l'UML (*Unified Modeling Language*) en tant que méthode de conception basée sur un ensemble de diagrammes. Nous avons utilisé le Diagramme de classes et le passage vers le modèle relationnel, diagramme d'objet ainsi que le diagramme de séquence.

IV.2-Approche du CKMS en tant que SecaaS

Dans cette section nous allons présenter l'architecture générale de la solution ainsi que le processus de création des clés de chiffrement qui résume les différentes étapes qui s'exécutent dans l'application.

IV.2.1-Architecteur de notre solution

La Figure 18 représente la solution du CKMS en tant que SecaaS basé sur le TPM en tant que base de confiance, l'architecture est composée de 4 couches qui sont :

1. La couche physique : Composante TPM
2. La couche virtualisation (*VMM*) : Composée par un ensemble de vTPM,
3. Couche des Virtuelles machines : Chaque une contient un système d'exploitation, et chaque système est rattaché à une instance de TPM (*vTPM*).
4. Couche CKMS as a SecaaS **:** C'est un ensemble d'instance de type service de sécurité. Chaque instance est dédiée à un groupe d'utilisateur (*entreprise*). Chaque instance utilise le TPM pour la création des clés via vTPM.

L'ensemble des couches représentent le service SecaaS dans le Cloud Computing.

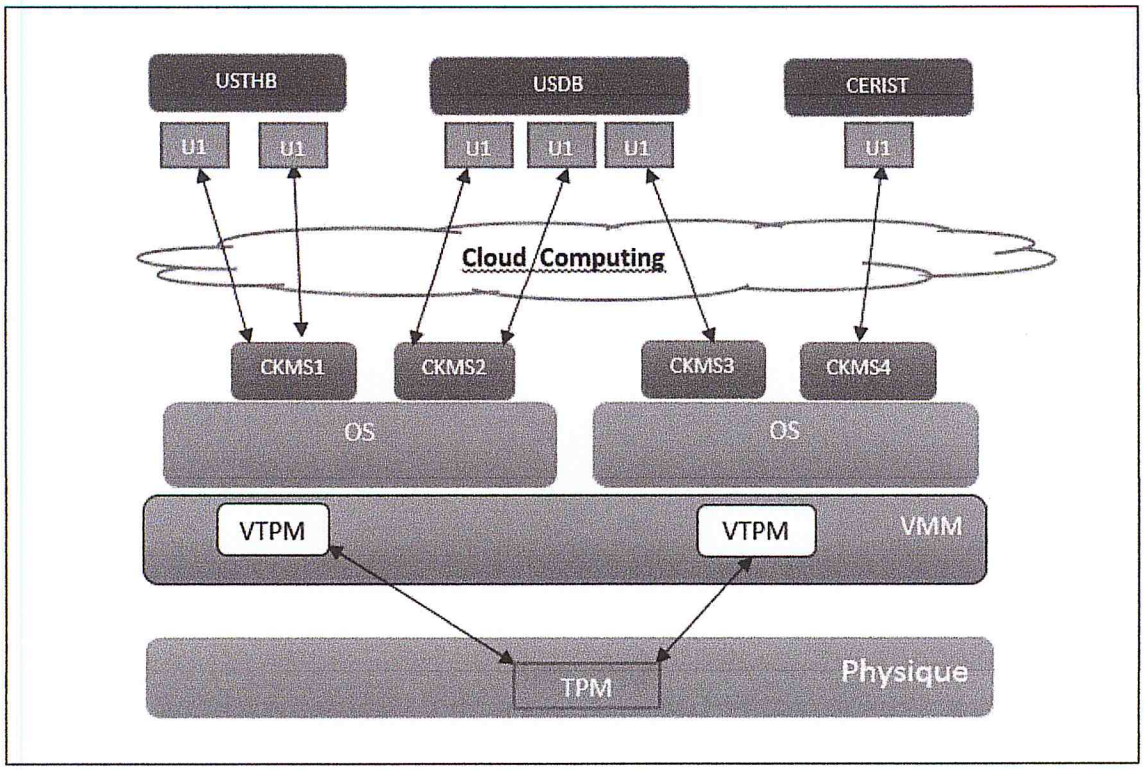


Figure 18: Architecture en couche du CKMS as a SecaaS

IV.2.2-Processus de création des Clés

Notre processus de création des clés (chiffrement / signature) est étroitement lié à l'architecture présentée dans la Figure-18.

Dans ce processus nous avons trois parties :

- L'utilisateur (*Le propriétaire de la clé*) ➔ noté U
- Le fournisseur du CKMS as a SecaaS ➔ noté C
- Le propriétaire du TPM (*Couche physique*) ➔ noté T

Chaque partie à une paire de clés (Publique, Privé), ces clés utilisées pour chiffrement/ déchiffrement ou signature : U (Pub_u , $Priv_u$), C (Pub_{ckms}, $Priv_{ckms}$) et T (Pub_{tpm}, $Priv_{tpm}$)

Le processus de création des clés est donne comme suit :

1- L'utilisateur envoyait une requête vers le ***CKMS*** pour lui créer une clé avec son clé Pub_u

2- CKMS envoyait une requête vers le ***TPM*** pour créer une clé pour l'utilisateur ***U***. La requête contient Pub_u et Pub_{ckms}.

3- TPM créer la clé, notée ***Key***

4- TPM signer la clé avec $Priv_{tpm}$, noté ***Key_Sign***

5- TPM chiffrer ***Key*** et ***Key_Sign*** avec Pub_u . Noté ***Encrypt_01***

6- TPM chiffrer ***Encrypt_01*** avec Pub_{ckms} . Noté ***Encrypt_02***

7- TPM envoyer ***Encrypt_02*** vers CKMS.

8- CKMS décrypte ***Encrypt_02*** avec ça clés privé $Priv_{ckms}$. Donc il à ***Encrypt_01***

9- CKMS envoyer ***Encrypt_01*** vers l'utilisateur.

10- L'utilisateur décrypte ***Encrypt_01*** avec sa clée privé $Priv_u$. Donc il à ***Key*** et ***Key_Sign.***

11- L'utilisateur vérifie ***Key_Sign*** par la clé publique de TPM, afin de vérifier l'authenticité de la clé ***Key***.

Les étapes du processus du 2eme jusqu'à le 7eme réalisés dans une seule session sécurisé, tel que TPM créer une session fermée dédié pour le CKMS, les commandes qui s'exécutent à l'intérieur de cette session sont chiffré, à la fin le TPM fermer la session .

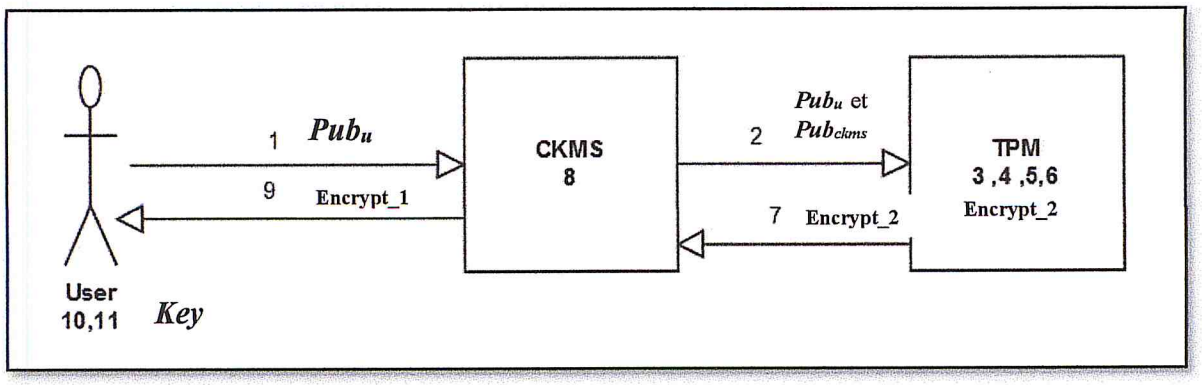


Figure 19 : Processus de création de la clé

IV.3-Modélisation

Il existe plusieurs méthodes utilisées dans la conception des solutions informatiques. Dans notre cas le choix retenu étant la méthode UML qui est la méthode la plus utilisée. Dans cette section nous allons présenter la méthode UML puis la partie conception, dans la conception nous avons utilisé trois diagrammes : diagramme de classes avec passage vers le modèle relationnel, digramme d'objets et enfin deux diagrammes de séquences.

IV.3.1- Diagramme de classes de notre solution

Suite à l'analyse des chapitres précédents on a présenté le diagramme de classes dans la Figure-16. A partir de ce modèle nous aurons :

- Une vue sur l'application.

- Les relations entre les différentes entités (*classes d'objets*) qui compose notre système.
- Les corps des différentes classes et on obtient une grande partie du code finale, ce qui nous mène vers la solution finale.

Les class sont les suivants :

- Une class key et ces 3 types : Symétrique, Asymétrique, RNG.
- Une class Algorithme
- Une class Owner
- Une class TPM-CKMS

La description de l'ensemble des méthodes est présentée dans le chapitre d'implémentation

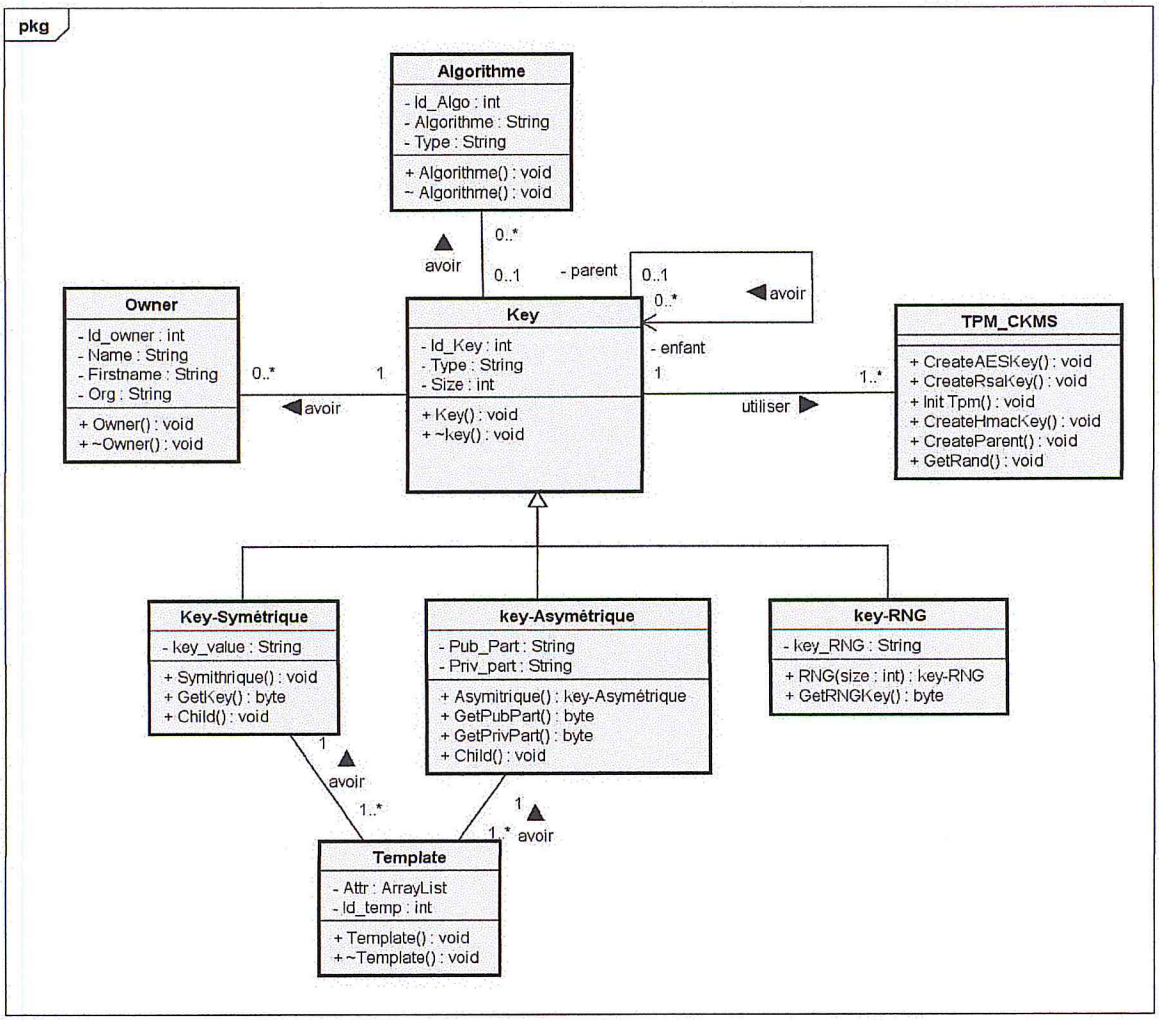


Figure 20: Diagramme de Classes de notre solution

IV.3.2 - Diagramme d'objets de notre solution

A partir de diagramme de classes on a construit le diagramme d'objets suivant qui montre un exemple de création de deux clés de différents types :

- Une de type Symétrique pour l'algorithme AES avec l'utilisation d'une Template T1.
- Une autre clé de type RNG, sans utilisation du Template.

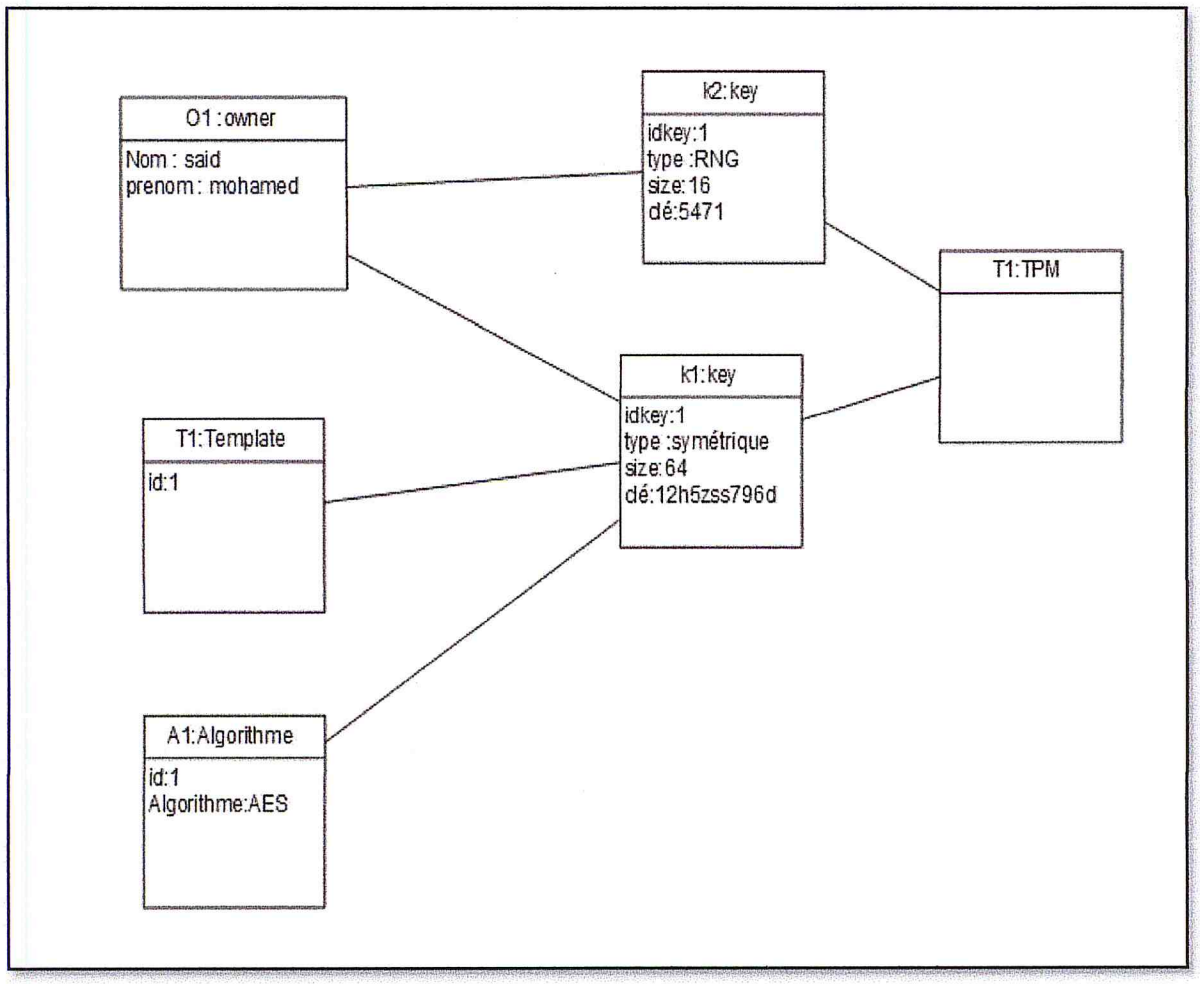


Figure 21: Diagramme d'objet de notre solution

IV.3.3 - Modèle Relationnel de notre DCKMS

À partir du diagramme de class présenté dans le Figure-20, et après application des règles de passage du diagramme de classes vers le modèle relationnel. Nous avons dans la figure 22 notre modèle relationnel :

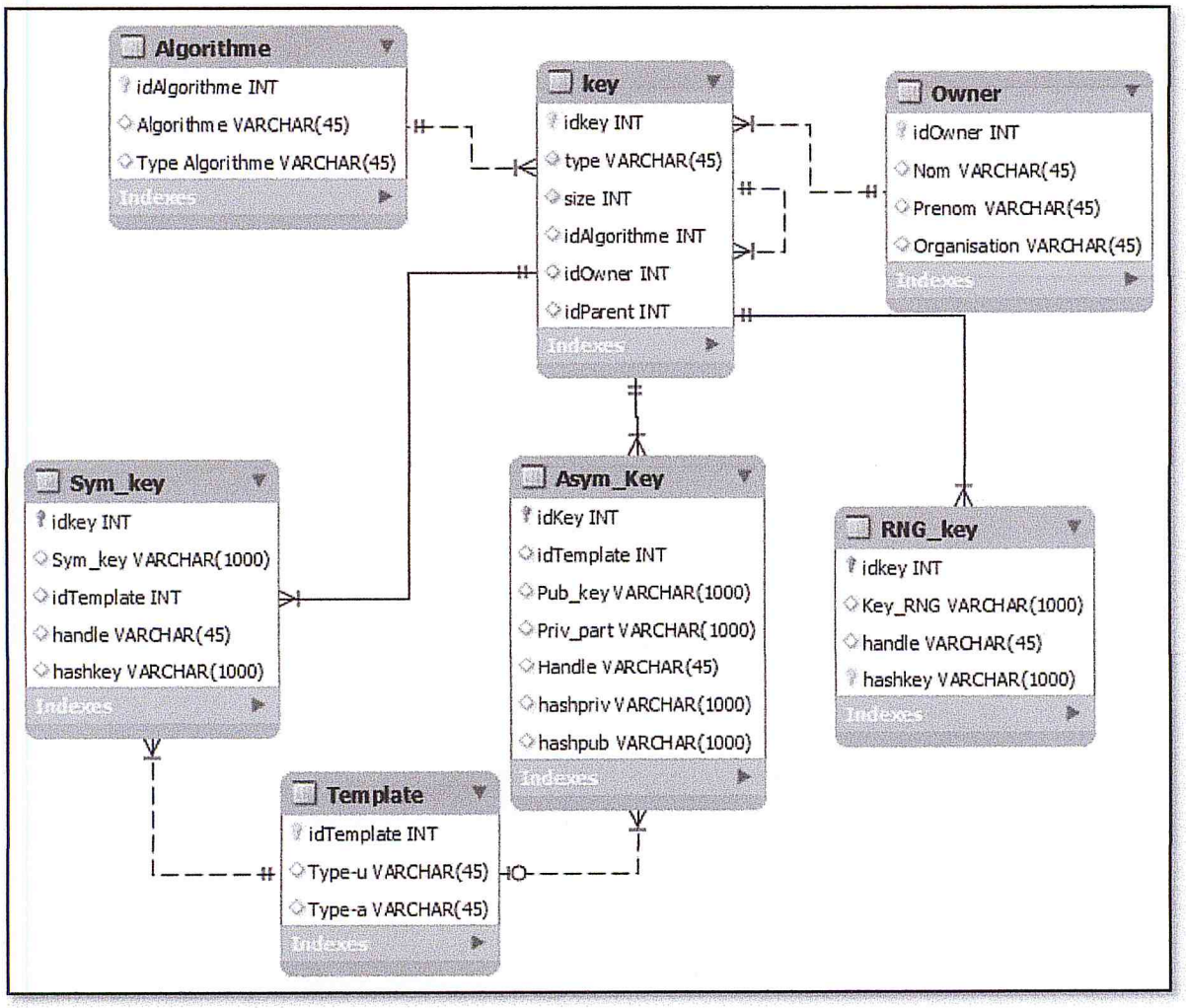


Figure 22: Modèle relationnel de notre DCKMS

Le modèle relationnel est composé de ces relations :

- **Key** (idkey , Type, size , #idAlgorithme , #idowner , # idtemplate , idparent)
- **Algorithme** (idAlgorithme, Algorithme,Type Algorithme)
- **Owner** (Idowner, Nom ,Prenom ,Organisation)
- **Template** (idTemplate, type-u,type-a)
- **Sym_key** (#idkey, #idTemplate, Sym_key,handle,hashkey)
- **Asy_key** (#idkey , #idTemplate , pub_key, priv_key ,Handle,hashpub,hashpriv)
- **RNG_key** (#idkey , keyRNG ,Handle,hashkey)

IV.3.4 – Diagrammes de Séquence de notre solution

Dans cette section nous présentons deux scénarios de création de clés modélisé par des diagrammes de séquences :

a. Création d'une clé

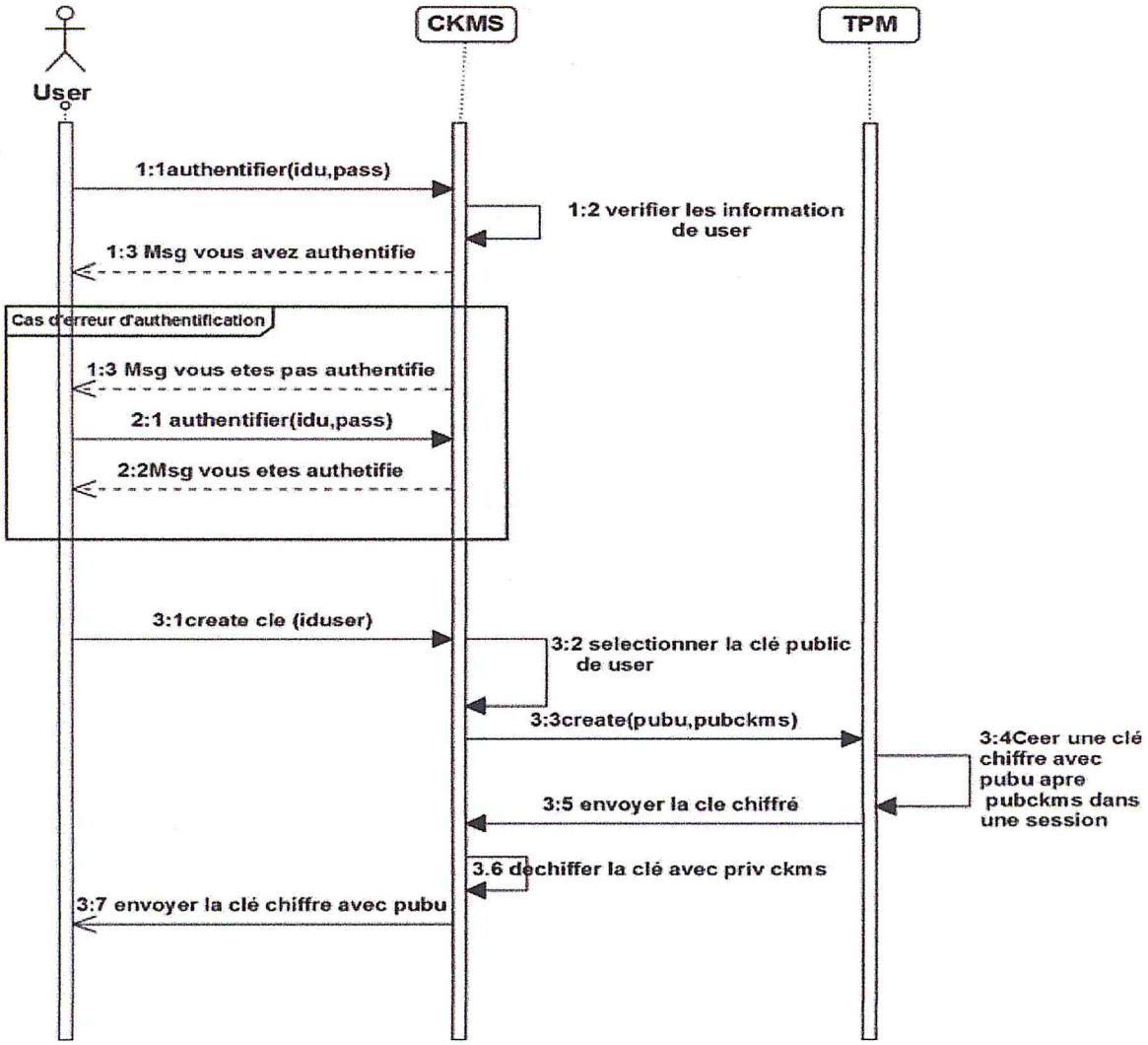


Figure 23: Diagramme de séquence de la création d'une clé

Ce diagramme représente le Scénario Général d'exécution de l'application :

- *1.1* L'utilisateur saisie ses information d'authentification
- *1 .2* le CKMS vérifié l'identité de l'utilisateur
- *1.3* Message de validation d'authentification
- *1.3, 2.1, 2.2* le cas où le client n'est pas authentifié
- *3.1* Le client demande la création d'une clé

- *3.2* le CKMS récupère la clé publique de l'utilisateur et envoyer la demande vers le TPM
- *3.3* le TPM créer la Clé, puis chiffre le avec la clé public de l'utilisateur, puis avec la clé publique de CKMS
- *3.4* le CKMS déchiffre la clé avec sa clé privée.
- *3.5* Envoyer la clé chiffrée avec la clé publique de l'utilisateur dans ce cas la clé reste confidentielle par ce que seul l'utilisateur qui peut la déchiffrer.

b. Création d'une clé enfant

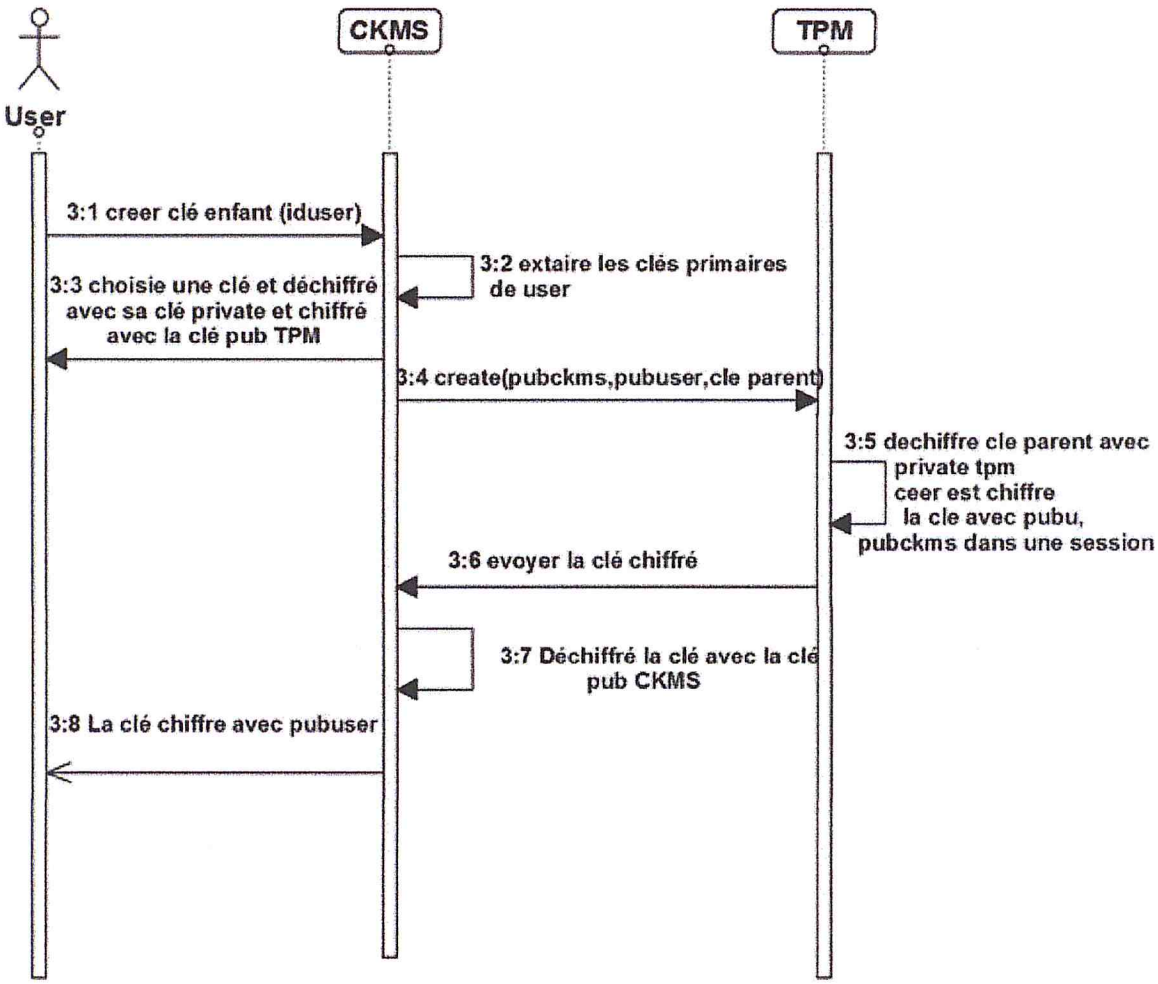


Figure 24 : Diagramme de Séquence de la création d'une clé enfant.

Ce diagramme représente la création d'une clé enfante à partir d'une clé père :

- *3.1* Envoyer la demande de la création
- *3.2* Le système CKMS extraire toutes clés primaires de l'utilisateur

- ***3.3*** L'utilisateur choisie une clé et le déchiffré avec sa clé public et chiffré avec la clé public de TPM
- ***3.4*** envoyer la demande vers le TPM
- ***3.5*** TPM déchiffré la clé avec sa clé privé et génère la clé enfant et le chiffré avec la clé publique de l'utilisateur puis avec la clé publique de CKMS et envoyer vers le CKMS
- ***3.6*** déchiffrer la clé avec sa clé privé
- ***3.7*** envoyer la clé vers l'utilisateur

IV.7 – Conclusion

Dans ce chapitre, nous avons présenté la conception de notre solution « CKMS en tant que SecaaS » de gestion de clés de chiffrement basé sur le TPM en tant que source de confiance pour la création des clés de chiffrement/signature.

En effet, le TPM peut créer des clés avec déférentes algorithme, Symétrique ou Asymétrique (*AES, RSA, RNG,ECC, HMAC*) et il peut signer les clés pour plus d'authenticité et confidentialité, le TPM est doté par une vaste liste d'algorithmes.

Afin de valider notre solution, nous avons choisi trois algorithmes à savoir : RSA : Asymétrique, AES : Symétrique, RNG et HMAC. L'implémentation de la solution sera décrite dans le chapitre suivant.

CHAPITRE V : SIMULATION ET RESULTATS

Chapitre V : Simulation et Résultat

V.1-Introduction :

Arrivé à ce stade nous pouvons nous estimer heureux, il ne reste qu'à commencer à écrire notre code en se basant sur l'analyse et la conception qu'on a obtenu dans les chapitres précédents et faire des tests sur les résultats de notre application.

V.2-Outils et environnement de développement

Nous avons utilisé des outils différents qui sont :

V.2.1-Microsoft Visual Studio (2017)

Microsoft Visual Studio est une suite de logiciels de développement pour Windows et mac OS conçue par Microsoft. La dernière version s'appelle Visual Studio 2017.

Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications web ASP.NET, des services web XML, des applications de bureau et des applications mobiles. Visual Basic, Visual C++ qui nous avons utilisé dans notre travail, Visual C# utilisent tous le même environnement de développement intégré (IDE), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du Framework .NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications web ASP et de services web XML grâce à Visual Web Développer. [27]

V.2.2-Simulateur TPM 2.0

Une spécification de bibliothèque de module de plateforme sécurisée publiée par le groupe TCG (*Trusted Computing Group*). Le résultat des scripts d'extraction est un ensemble complet de fichiers sources pour un simulateur TPM (*Trusted Platform Module*) 2.0, qui fonctionne sous Windows, Linux, ainsi que Genode (en appliquant les correctifs appropriés). [28]

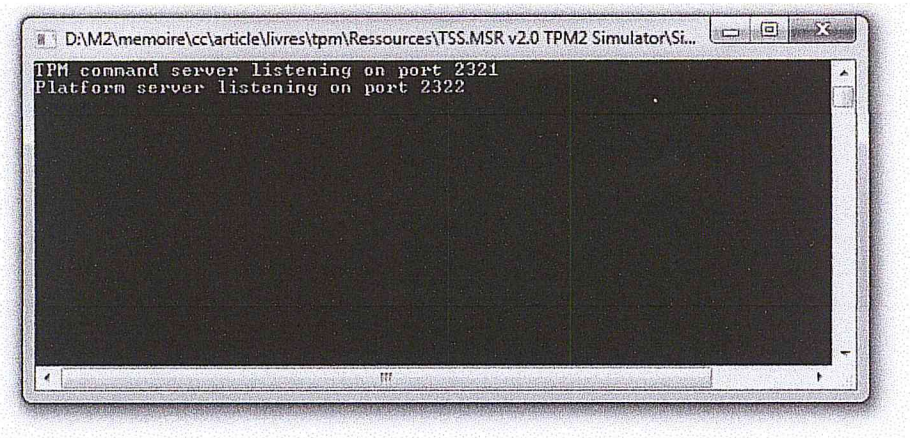


Figure 25 : Simulateur TPM2.0

V.2.3-Bibliothèques de projet TSS.MSR

TSS.Net et TSS.C ++ simplifient l'écriture d'applications Windows utilisant TPM 2.0. Ces bibliothèques fournissent un accès de bas niveau au TPM et traitent la plupart des problèmes complexes qui surviennent lors de l'interaction avec le TPM. La bibliothèque TSS.Net est écrite en code managé (C #) et peut être utilisée sur les systèmes Windows 8+ par n'importe quelle application gérée. TSS.C ++, également appelé TSS.CPP, est écrit en C ++ et fournit la même fonctionnalité. En plus de prendre en charge l'accès à un TPM physique, les bibliothèques TSS.MSR peuvent également être connectées à un simulateur TPM pour activer le développement et le débogage d'applications sur des plates-formes qui ne disposent pas d'un périphérique TPM 2.0. La connexion au simulateur se fait via un socket TCP / IP. [27]

V.2.4 -XAMPP

XAMPP est un ensemble de logiciels permettant de mettre en place facilement un serveur Web local, un serveur FTP et un serveur de messagerie électronique. Il s'agit d'une distribution de logiciels libres (*X (cross) Apache MariaDB Perl PHP*) offrant une bonne souplesse d'utilisation, réputée pour son installation simple et rapide. Ainsi, il est à la portée d'un grand nombre de personnes puisqu'il ne requiert pas de connaissances particulières et fonctionne, de plus, sur les systèmes d'exploitation les plus répandus. [29]

V.2.5 -L'interface PHPmyadmin

PHPMyAdmin est une application web qui permet de gérer un serveur de bases de données MySQL. Dans un environnement multiutilisateur, cette interface écrite en PHP permet également de donner à un utilisateur un accès à ses propres bases de données. [30]

V.2.6 - Serveur apache

C'est le serveur le plus répandu sur Internet, permettant la configuration de l'environnement d'exécution de pages web. Il s'agit d'une application fonctionnant à la base sur les systèmes d'exploitation de type Unix, mais il a désormais été porté sur de nombreux systèmes, dont Microsoft Windows grâce à sa conception modulaire (*morceaux de code*) qui correspond à différents aspects ou fonctions du serveur. Cette conception autorise le développeur à choisir quelles fonctionnalités seront incluses dans le serveur en sélectionnant les modules à charger soit à la compilation, soit à l'exécution. Elle lui permet aussi d'écrire son propre morceau de code qui pourra ensuite être facilement intégré dans le serveur Web Apache. [29]

V.2.7 -Langages d'implémentation utilisés

Les langages utilisés sont :

- Langage C++ : Le C++ est le descendant du langage C. Ces deux langages, bien que semblables au premier abord, sont néanmoins *différents*. Le C++ propose de nouvelles fonctionnalités, comme la programmation orientée objet (POO). Elles en font un langage très puissant qui permet de programmer avec une approche différente du langage C. [31]
- Langage PHP (Hypertext Preprocessor) : C'est un langage compilé (*à partir de la version 5*) et exécuté du côté serveur (*comme les scripts de CGI, ASP,...*) et non du côté client, un client écrit en JavaScript ou une applet Java s'exécute sur votre ordinateur. La syntaxe du langage provient de celles du langage C, du Perl et du Java. [31]
- Langage de requête SQL : Pour Communiquer avec une base de données, on a besoin de lui envoyer des commandes ou instructions appelées requêtes. Que ce soit pour la création, la suppression d'une table, la modification, l'insertion ou la sélection de données, le langage standard de requêtes est SQL. SQL ou (*Standard Query Language)* est un langage permettant d'interroger les bases de données de manière simple. Il est doté d'une syntaxe particulière que l'on doit respecter pour que la communication avec la base se passe au mieux. Son succès est dû essentiellement à sa simplicité et au fait qu'il énonce des requêtes en laissant le SGBD responsable de la stratégie d'exécution.

V.3-Archeticture Global de la Solution

Notre solution est basée sur une approche multi-tiers. Nous avons : Le TPM, le CKMS et les utilisateurs (Figure 25).

- TPM : C'est le simulateur qui remplace deux couches : TPM (physique) et le vTPM (VMM)
- CKMS (*en tant que SecaaS*) : Est composé par trois principales composantes :
  - Ensemble d'API qui se charge des connexions avec le TPM (*simulateur*), l'accès à la base de données et la partie présentation (*serveur web*)
  - Dictionnaire de données (*D-CKMS*) : Une base de données gérée par un SGBD (*MySQL*), contient les clés avec les informations en relation.
  - Interface web : Et Grâce aux langages HTML, CSS3 on a construit une interface de simulation, et en utilisant le langage PHP et MySQL on a exécuté des requêtes SQL qui vont s'exécuter via le serveur Xampp (Figure 26)
- Client : Explorateur internet explorer ou autre application mobile.

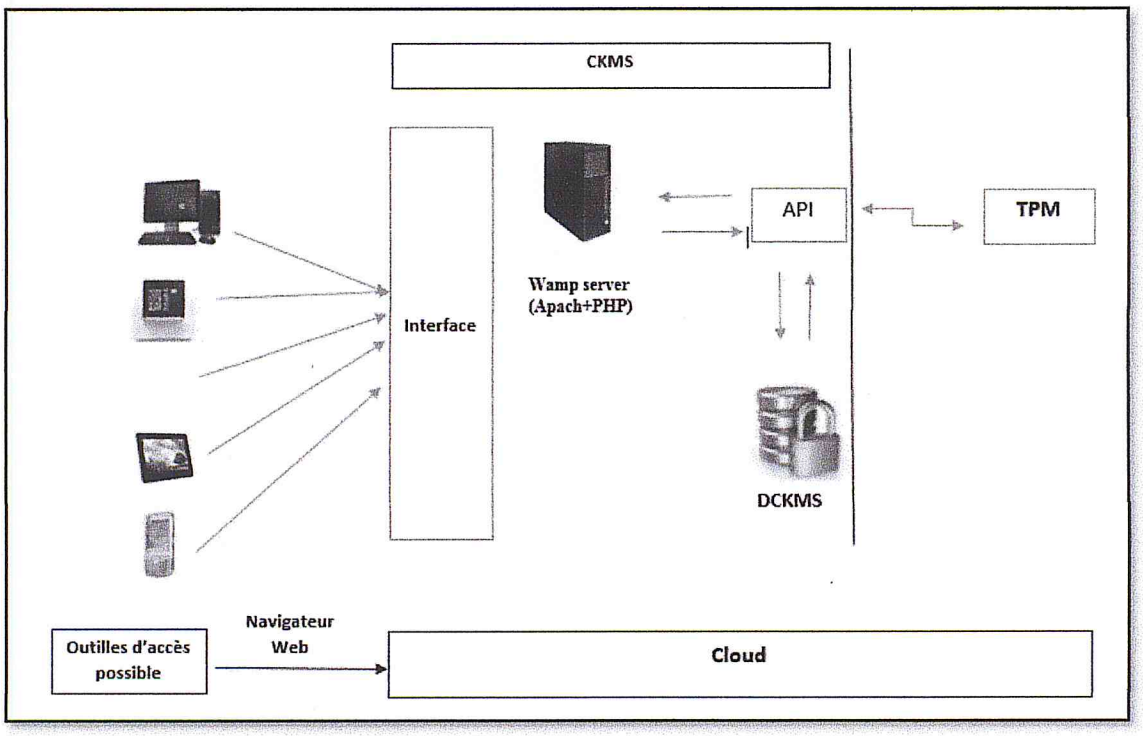


Figure 26: Schéma Global de la Solution

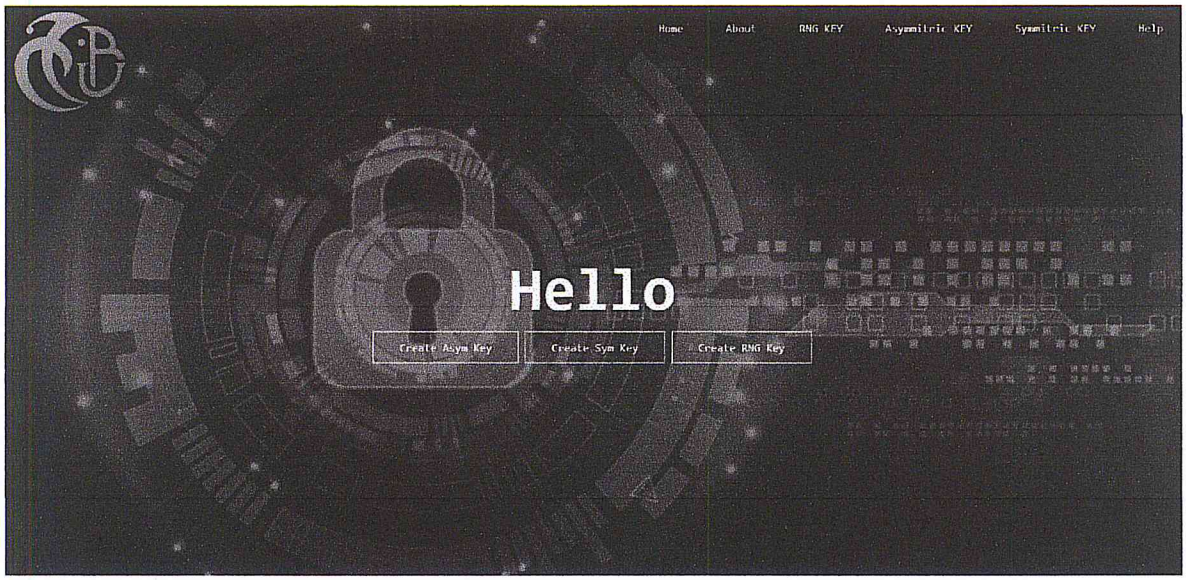


Figure 27: Interface principale du CKMS en tant SecaaS

V.4-Simulation

Dans notre solution on remplacer le TPM (*Couche physique*) et le vTPM (*Couche virtualisation*) par le simulateur TPM2.0.

V.4.1-Clé RNG

A travers le formulaire simple suivant en peut créer une clé RNG d'âpre des valeurs aléatoires

Figure 28: formulaire de clé RNG

Voilà la partie de code source de la classe RNG

```
Key_RNG::Key_RNG(int size, string type, Owner o, Algo alg) :Key(size, type, o, alg)
{
    int id;
    PolicyTree p(PolicyCommandCode(TPM_CC::PolicyCpHash, ""));
    TPMT_HA policyDigest = p.GetPolicyDigest(TPM_ALG_ID::SHA1);
    AUTH_SESSION session  = tpm.StartAuthSession(TPM_SE::POLICY, TPM_ALG_ID::SHA1);
    p.Execute(tpm, session);
    tpm.keycryptage = tpm.RSAEncrypte();
    tpm.keycrypt = tpm.keycryptage.objectHandle;

    key_RNG = tpm.GetRandom(size);
    cout <<"la clé RNG en claire "<< getKeyRNG() << endl;
    vector<BYTE> data = vector<BYTE>{ getKeyRNG() };
    TPMT_HA dataToSign = TPMT_HA::FromHashOfData(TPM_ALG_ID::SHA1, data);
    //cout << "\n" << dataToSign.digest << endl;
    auto enc = tpm.RSA_Encrypt(tpm.keycrypt, data, TPMS_NULL_ASYM_SCHEME(), tpm.NullVec);
    //cout << "\n RSA-encrypted data: " << enc << endl;
     auto dec = tpm.RSA_Decrypt(tpm.keycrypt, enc, TPMS_NULL_ASYM_SCHEME(), tpm.NullVec);
    //cout << "\n decrypted data: " << dec << endl;
```

Pour créer une clé RNG on doit d'abord créer deux objets qui sont un objet Owner et un objet Algorithme pour chaque objet on doit remplir des informations spéciales.

```
Owner o(1,"saidia","mohamed","CERIST");
Algo a(71, "RNG", "Null");
Key_RNG key(10, "RNG", o, a);
```

On doit hacher ce résultat et crypter avec la clé publique de l'utilisateur pour garantir la confidentialité et l'intégrité de l'information et même temps on va crypter la clé avec la clé public user.

Voilà la clé dans l'interface user :

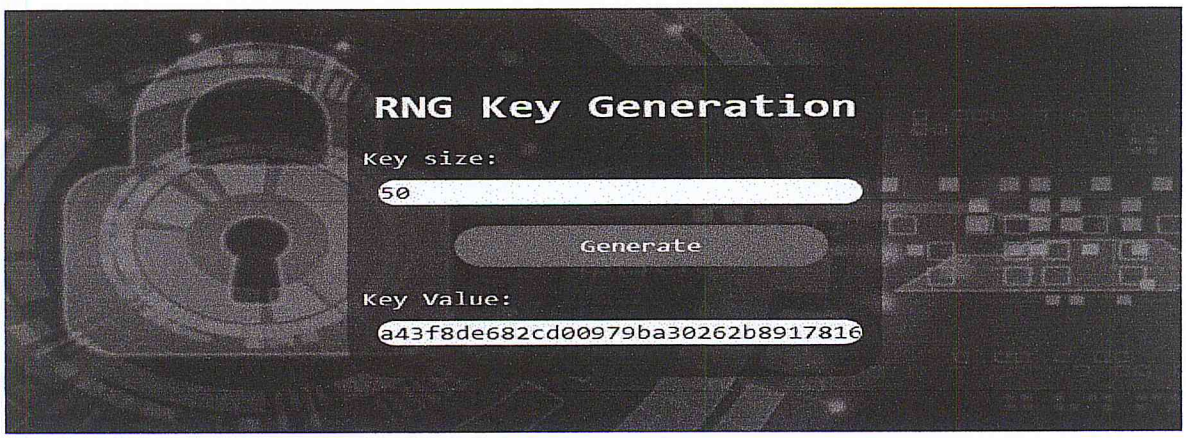


Figure 29: la clé RNG dans l'interface user

La clé dans le Dictionnaire CKMS.

	idKey	cle	hachkey
Modifier Copier Effacer	37	8a23bc07dca80abaadd9f97d94d5f3a690538554a7d8e2f954...	0e64590d4187d2522e26f816b876d620c57ab19965fe1de695...
Modifier Copier Effacer	38	63acd0980d9494f7c16d3602a17f57cc1619de819b33714aa2...	716b4247335d7d3450dd76f661dbaa5e4eecce7e3633dec495...
Modifier Copier Effacer	39	4a3d77406e9801cbe9ec05c1ce1cfb628393afa28cfe5fc0b6...	65514e710e6a8a5b99fd5479cccaa19443779f3ecc5a9cf33c...
Modifier Copier Effacer	41	69e3126ecce81e5d4776af5918c21e0d22a7b8f726742d8071...	2a0c036eafde3797e09a3157c24409749f680a21d959d00c23...

V.4.2-Clé symétrique (*AES*)

Pour créer une clé AES on doit remplir le formulaire suivant :

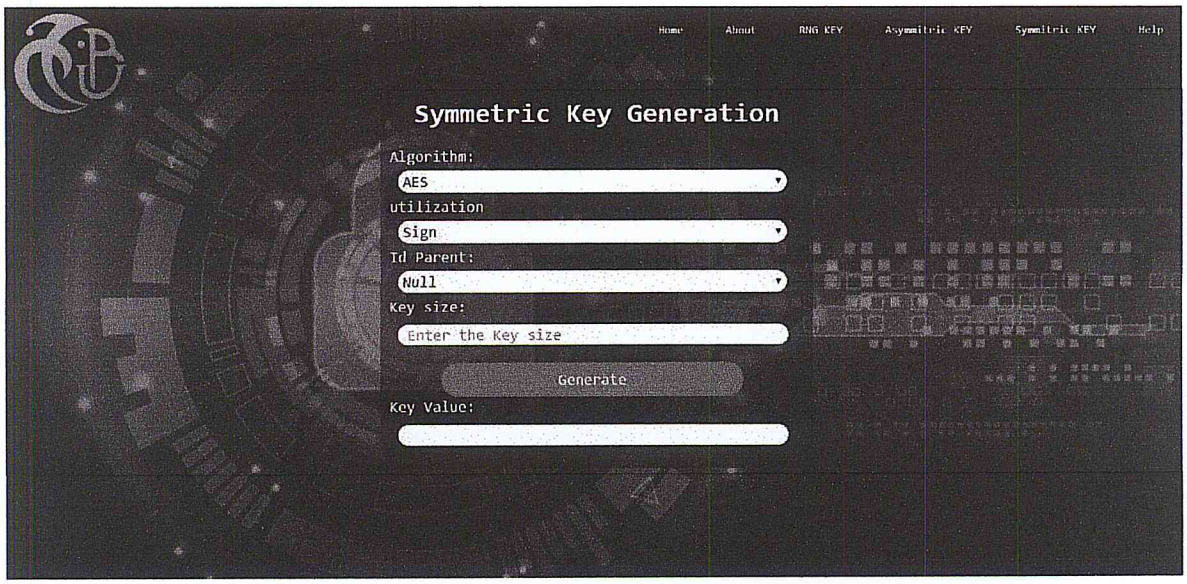


Figure 30: formulaire de clé Asymétrique

Voilà une un parti de code source de la class symétrique

```
Key_Sym::Key_Sym(int size,string type,Template T,Owner o,Algo alg):Key(size,type,o,alg)
{
    this->tmplt = T;
    int id;

    PolicyTree p(PolicyCommandCode(TPM_CC::PolicyCpHash, ""));
    TPMT_HA policyDigest = p.GetPolicyDigest(TPM_ALG_ID::SHA1);
    AUTH_SESSION session = tpm.StartAuthSession(TPM_SE::POLICY, TPM_ALG_ID::SHA1);
    p.Execute(tpm, session);

    tpm.keycrypt = tpm.RSAEncrypte().objectHandle;
    if (type == "storagekey") {
        Key_Value = tpm.CreateRSAparent().outPublic.unique->ToBuf();
        cout << getKeyValue() << endl;
        int handle = tpm.handleparent.handle;
        string pk = tpm.BytesToStr(getKeyValue());
        vector<BYTE> data = vector<BYTE>{ getKeyValue() };
        TPMT_HA dataToSign = TPMT_HA::FromHashOfData(TPM_ALG_ID::SHA1, data);
        string hpk = tpm.BytesToStr(dataToSign.digest);
        auto enc1 = tpm.RSA_Encrypt(tpm.keycrypt, dataToSign.digest, TPMS_NULL_ASYM_SCHEME(), tpm.NullVec);
        //cout << "\n" << enc1 << endl;
```

Pour créer une clé AES on doit d'abord créer trois objets qui sont un objet Owner, un objet Algorithme et la Template qui été importante dans la création de la clé qui sa diffère à l'aide de son type d'utilisation (Sign ,Decrypt , Encrypt) , pour chaque objet on doit remplir des informations spéciales .

```
Owner o(1,"saidia","mohamed","CERIST");
Algo a(71, "AES", "Symetrique");
Template t( 1,a.getTypeAlgo(),"decryp");
Key_Sym s(128, "sign", t, o, a);
```

Voilà la clé dans l'interface utilisateur :

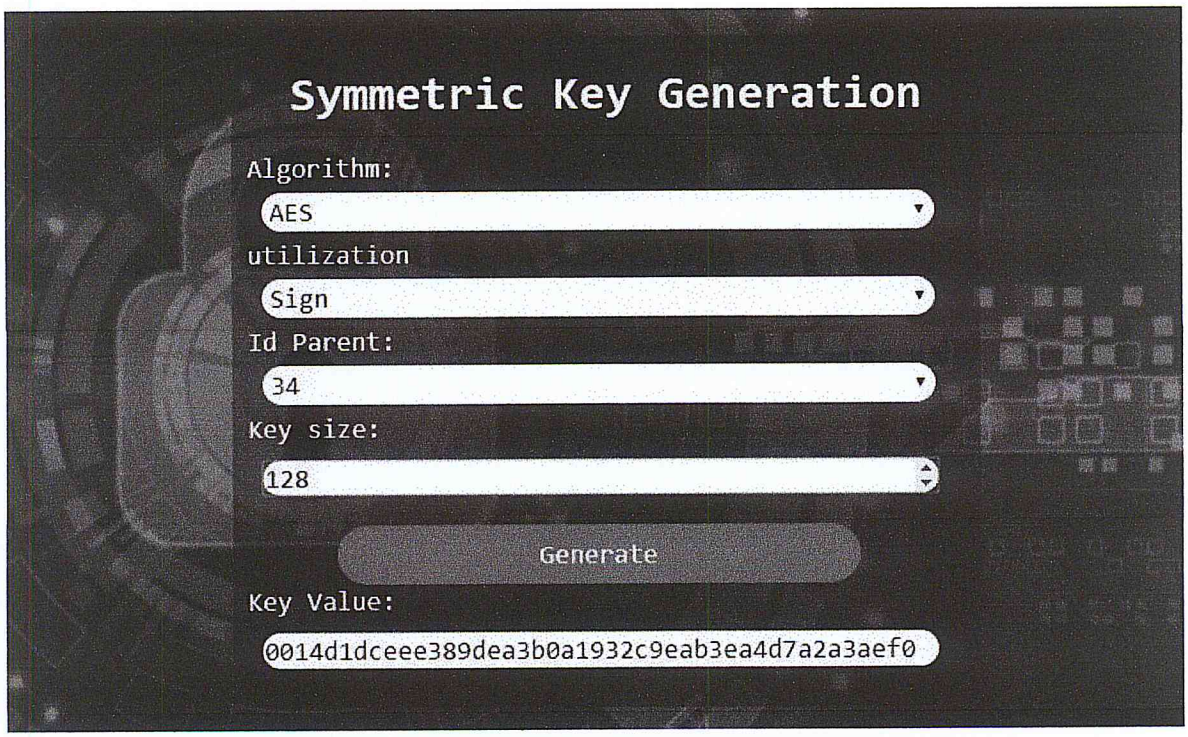


Figure 31: Clé AES dans l'interface user

La clé dans le Dictionnaire CKMS :

Dans la table on a insérée la clé âpres handle ou bien l'identifiant de la clé et le hach de la clé cryptée avec la clé publique de user

idkey	Sym_key	idTemplate	Handle	hachkey
34	0100b62ccb3718b96feb4e355c60d3b5aabb944e0f907d0ebc...	1	-2147483648	b6745eeb2217ebe575a9df5562afb39b7a5491ad50be1d1087...
35	0100b62ccb3718b96feb4e355c60d3b5aabb944e0f907d0ebc...	1	-2147483648	5f2e84ab23ac368b4b758a9ede03a7851272d1dc31bef5c833...
36	0100b62ccb3718b96feb4e355c60d3b5aabb944e0f907d0ebc...	1	-2147483647	67221ffa32f49ac2d0cf3c1697629c10e1f4fb4866019cadca...
53	769cdb3089a52eb6560d6f1984460840fadc531ab242882f54...	1	-2147483647	899830e75873c3bd3b90662077934c698248fbe3999146fb33...
54	658ea34dc9d96a755685f34ce135feab700a319ecabd7ea666...	1	-2147483647	7225c696bd2fa20a3f90a19c1e77ab130dbb559166c83aaa90...

V.4.3-Clé Asymétrique (RSA)

Pour créer une clé Asymétrique on doit remplir le formulaire suivant :

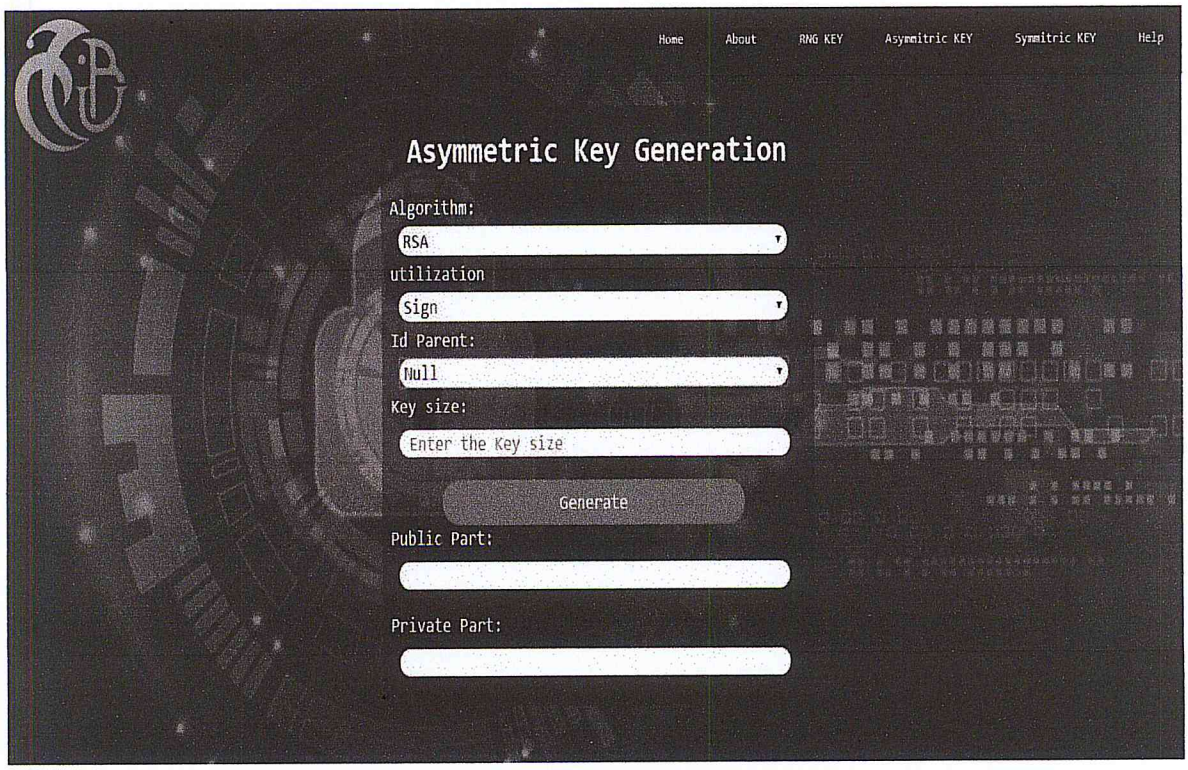


Figure 32: le formulaire de RSA

Voilà une partie de code source de la classe asymétrique :

```
CreateResponse k = tpm.CreateRSAdecrypteencrypte(type, size);
//cout << k.outPrivate.ToBuf();
Priv_Part = k.outPrivate.buffer;
Pub_Part = k.outPublic.unique->ToBuf();

cout << "la partie public :" << Priv_Part << endl;
cout << "la partie private :" << Pub_Part << endl;
auto enc1 = tpm.RSA_Encrypt(tpm.keycrypt, getPriv(), TPMS_NULL_ASYM_SCHEME(), tpm.NullVec);
auto dec1 = tpm.RSA_Decrypt(tpm.keycrypt, enc1, TPMS_NULL_ASYM_SCHEME(), tpm.NullVec);
//cout << enc3<<endl;
//cout <<"dec"<< dec3 << endl;
vector<BYTE> data1 = vector<BYTE>{ getPriv() };
TPMT_HA dataToSign1 = TPMT_HA::FromHashOfData(TPM_ALG_ID::SHA1, data1);
auto enc2 = tpm.RSA_Encrypt(tpm.keycrypt, dataToSign1.digest, TPMS_NULL_ASYM_SCHEME(), tpm.NullVec);
auto dec2 = tpm.RSA_Decrypt(tpm.keycrypt, enc2, TPMS_NULL_ASYM_SCHEME(), tpm.NullVec);
vector<BYTE> data2 = vector<BYTE>{ getPub() };
TPMT_HA dataToSign2 = TPMT_HA::FromHashOfData(TPM_ALG_ID::SHA1, data2);
auto enc3 = tpm.RSA_Encrypt(tpm.keycrypt, dataToSign2.digest, TPMS_NULL_ASYM_SCHEME(), tpm.NullVec);
```

Pour créer une clé RSA on doit d'abord créer trois objets qui sont un objet Owner et un objet Algorithme et la Template qui été importante dans la création de la clé qui sa défaire à l'aide de type d'utilisation (*Sign* ,*Decrypt* , *Encrypt*) , pour chaque objet on doit remplir des informations spéciales :

```
Owner o(1,"saidia","mohamed","CERIST");
Algo a(71, "RSA", "Asymétrique");
Template t( 1,a.getTypeAlgo(),"a");
Key_Asym k(1024, "decrypt", t, o, a);
```

Voilà la clé RSA dans l'interface user :

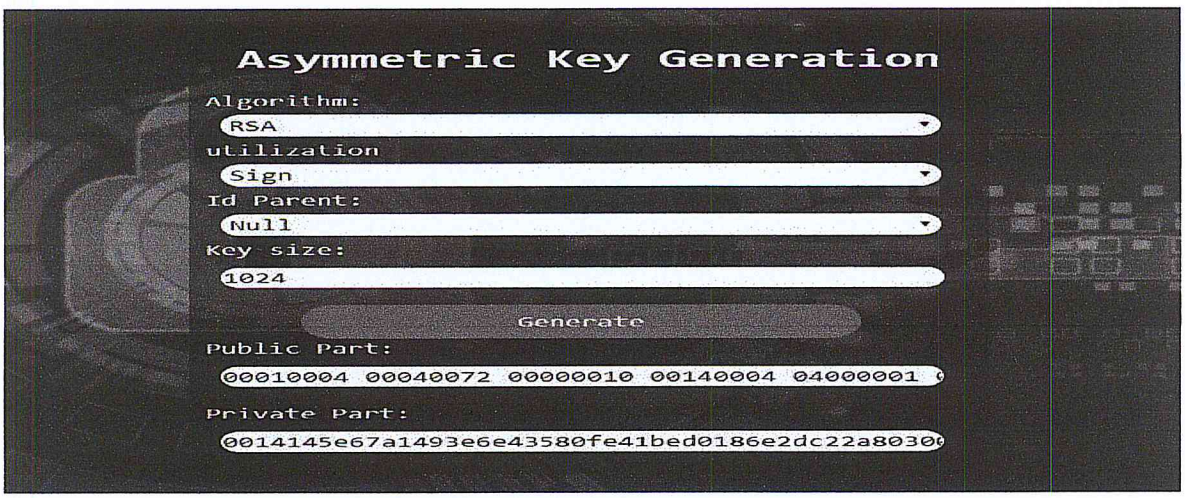


Figure 33: une clé RSA dans l'interface user

La clé dans le Dictionnaire CKMS :

Dans la table on a inséré la clé âpre handle ou bien l'identifiant de la clé et le hach de la clé crypté avec la clé publique de user

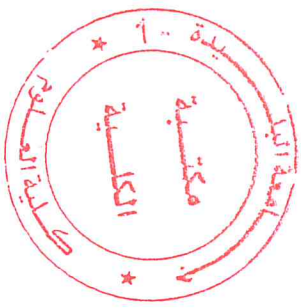
idKey	idTemplate	Pub_key	Priv_part	hachpub_key	hachpriv_key	Handle
55	1	0100e631447902881015138(	0e5c0c7178118a300	9c4380074555e7ca5cafc7d1e741t	2c935af5b5c2656ec3f6	-2147483646
57	1	0100afd0fad64cb96f2660dde	935f4820822f0ef08a(	7375c7ad71125016e1b65b60e386	3ed81178962d3d7f67f3	-2147483646

V.4.4-Clé parent (*Storage Key*)

Storage key ou bien clé parent est une clé juste pour la création des sous clés d'après une valeur spécifiée qui sa diffère de la valeur de semence de TPM.

Une partie de code :

```
if (type == "storagekey") {
    Key_Value = tpm.CreateRSAparent().outPublic.unique->ToBuf();
    cout << getKeyValue() << endl;
    int handle = tpm.handleparent.handle;
    string pk = tpm.BytesToStr(getKeyValue());
    vector<BYTE> data = vector<BYTE>{ getKeyValue() };
    TPMT_HA dataToSign = TPMT_HA::FromHashOfData(TPM_ALG_ID::SHA1, data);
    string hpk = tpm.BytesToStr(dataToSign.digest);
    auto enc1 = tpm.RSA_Encrypt(tpm.keycrypt, dataToSign.digest, TPMS_NULL_ASYM_SCHEME(), tpm.NullVec);
    //cout << "\n" << enc1 << endl;
    auto dec1 = tpm.RSA_Decrypt(tpm.keycrypt, enc1, TPMS_NULL_ASYM_SCHEME(), tpm.NullVec);
    //cout << "\n" << dec1 << endl;
    string hach = tpm.BytesToStr(enc1);
    try
    {
        sql::Connection *cnx;
        sql::Driver *driver;
        //sql::PreparedStatement *prs;
        sql::ResultSet *res;
        //sql::Statement *stmt;
        sql::PreparedStatement *prs;
        driver = get_driver_instance();
```



Pour créer un Storage key on doit d'abord créer trois objets qui sont un objet Owner et un objet Algorithme et une Template, pour chaque objet on doit remplir des informations spéciales

```
Owner o(1,"saidia","mohamed","CERIST");
Algo a(71, "Null", "Null");
Template t( 1,a.getTypeAlgo(),"a");
Key_Asym k(1024, "storagekey", t, o, a);
```

Voila la clé storage key(clé parent) en RSA dans l'interface d'utilisateur :

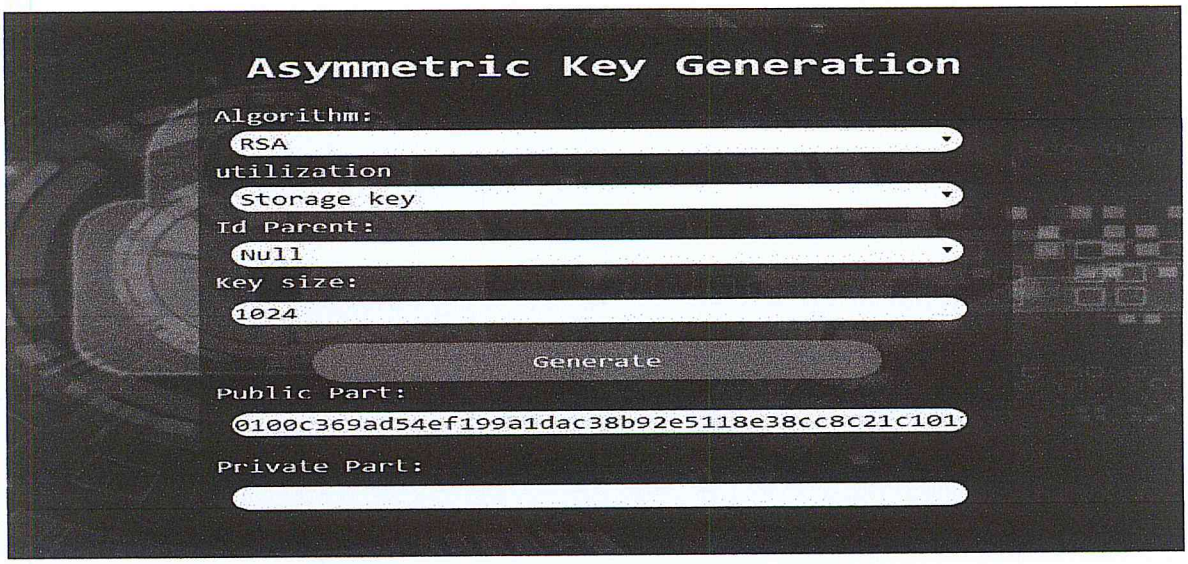


Figure 34: La clé parent RSA dans l'interface user

La clé dans le Dictionnaire CKMS :

Dans la table en a insérer la clé âpre handle ou bien l'identifiant de la clé et le hach de la clé crypté avec la clé publique de user.

658ea34dc9d96a755685f34ce135feab700a319ecabd7ea666...	1	-2147483647	7225c696bd2fa20a3f90a19c1e77ab130dbb559166c83aaa90...
01008457b6e54e82ae03e16bb52aaa41e0de5c52cf92bd44c2...	1	-2147483648	44fe49c847bdf8ba083a12fb20343b80f9f0c9bb2655600ff2...

V.4.5 Clé de signature(*HMAC*)

Cette clé est spécialement conçue pour utiliser dans l'algorithme HMAC

- Voilà une partie de code source de la fonction HMAC :

```
TPMT_PUBLIC templ(TPM_ALG_ID::SHA256, TPMA_OBJECT::sign | TPMA_OBJECT::fixedParent |
    TPMA_OBJECT::fixedTPM | TPMA_OBJECT::userWithAuth, NullVec,
    TPMS_KEYEDHASH_PARMS(TPMS_SCHEME_HMAC(hashAlg)),
    TPM2B_DIGEST_Keyedhash(NullVec));

// The key is passed in in the SENSITIVE_CREATE structure
TPMS_SENSITIVE_CREATE sensCreate(NullVec, key);
vector<TPMS_PCR_SELECTION> pcrSelect;

// "Create" they key based on the externally provided keying data
CreatePrimaryResponse newPrimary = CreatePrimary(_AdminOwner,
    sensCreate,
    templ,
    NullVec,
    pcrSelect);
```

Pour créer un HMAC key on doit d'abord créer trois objets qui sont un objet Owner et un objet Algorithme et une Template, pour chaque objet on doit remplir des informations spéciales

```
Owner o(1,"saidia","mohamed","CERIST");
Algo a(71, "HMAC", "Null");
Template t( 1,a.getTypeAlgo(),"a");
Key_Sym s(128, "sign", t, o, a);
```

Voila la clé Hmac dans l'interface d'utilisateur :

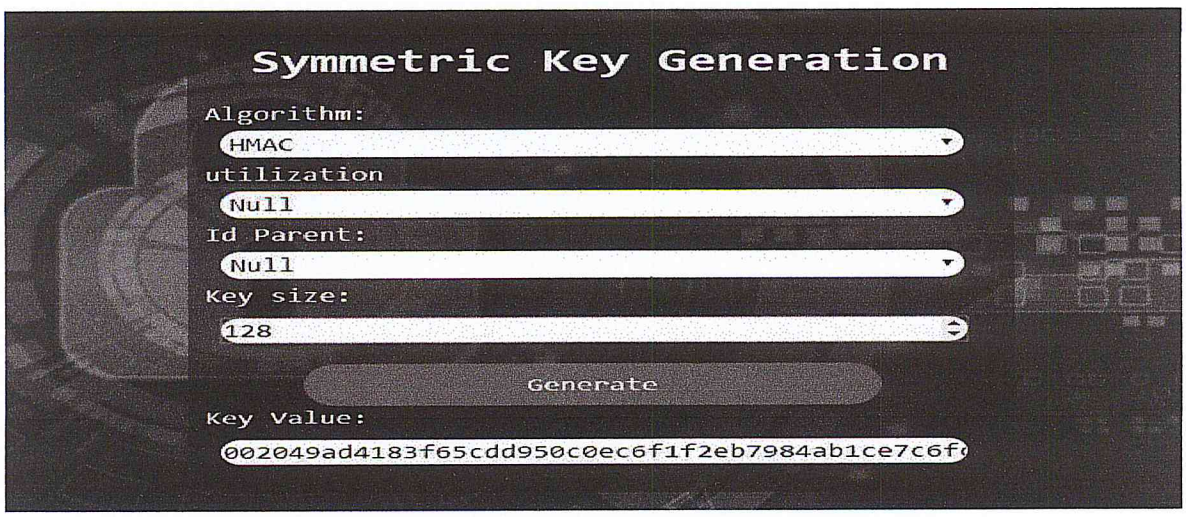


Figure 35: La clé HMAC dans l'interface user

La clé dans le Dictionnaire CKMS :

Dans la table on a inséré la clé âpres handle ou bien l'identifiant de la clé et le hach de la clé cryptée avec la clé publique de user

idkey	Sym_key	idTemplate	Handle	hachkey
34	0100b62ccb3718b96feb4e355c60d3b5aabb944e0f907d0ebc...	1	-2147483648	b6745eeb2217ebe575a9df5562afb39b7a5491ad50be1d1087...
35	0100b62ccb3718b96feb4e355c60d3b5aabb944e0f907d0ebc...	1	-2147483648	5f2e84ab23ac368b4b758a9ede03a7851272d1dc31bef5c833...
36	0100b62ccb3718b96feb4e355c60d3b5aabb944e0f907d0ebc...	1	-2147483647	67221ffa32f49ac2d0cf3c1697629c10e1f4fb4866019cadca...
53	769cdb3089a52eb6560d6f198446084Ofadc531ab242882f54...	1	-2147483647	899830e75873c3bd3b90662077934c698248fbe3999146fb33...
54	658ea34dc9d96a755685f34ce135feab700a319ecabd7ea666...	1	-2147483647	7225c696bd2fa20a3f90a19c1e77ab130dbb559166c83aaa90...

V.4.6-Clé enfant (*Child key*)

Cette clé créé à partir d'une clé parent (*Storage Key*) on peut créer des Child keys symétrique ou bien asymétrique selon le type d'utilisation , pour les Child key Asymétrique (RSA) en peut créer des clé de signatures ou bien de cryptage ou bien décryptage selon le besoin ,pour les clés symétriques (AES) on peut créer des clés de signature ou bien des clés de cryptage Voilà une partie de code :

```
void  Key_Sym::childkey(int ido, int idt, int ida,int size,string type) {
    PolicyTree p(PolicyCommandCode(TPM_CC::PolicyCpHash, ""));
    TPMT_HA policyDigest = p.GetPolicyDigest(TPM_ALG_ID::SHA1);
    AUTH_SESSION session = tpm.StartAuthSession(TPM_SE::POLICY, TPM_ALG_ID::SHA1);
    p.Execute(tpm, session);
    int t[100], i, id, idk;
    int had;
    try
    {
        sql::Connection *cnx;
        sql::Driver *driver;
        //sql::PreparedStatement *prs;
        sql::ResultSet *res;
        //sql::Statement *stmt;
        sql::PreparedStatement *prs;
        driver = get_driver_instance();
        cnx = driver->connect("tcp://127.0.0.1:3306", "root", "");
        cnx->setSchema("dckms");
        prs = cnx->prepareStatement("SELECT idkey from cle where `type-u`='storagekey' and idOwner=?");
        prs->setInt(1, ido);
        res = prs->executeQuery();
        i = 0;
```

Pour créer une clé Child il faut créer en premier une clé Storage key âpres, lancer la fonction de création de Child comme suit :

```
Owner o(1,"saidia","mohamed","CERIST");
Algo a(71, "AES"," Symetrique");
Template t( 1,a.getTypeAlgo(),"a");
Key_Sym k;
k.childkey(1, 1, 71,128,"decrypt");
```

Voilà la clé child dans l'interface user :

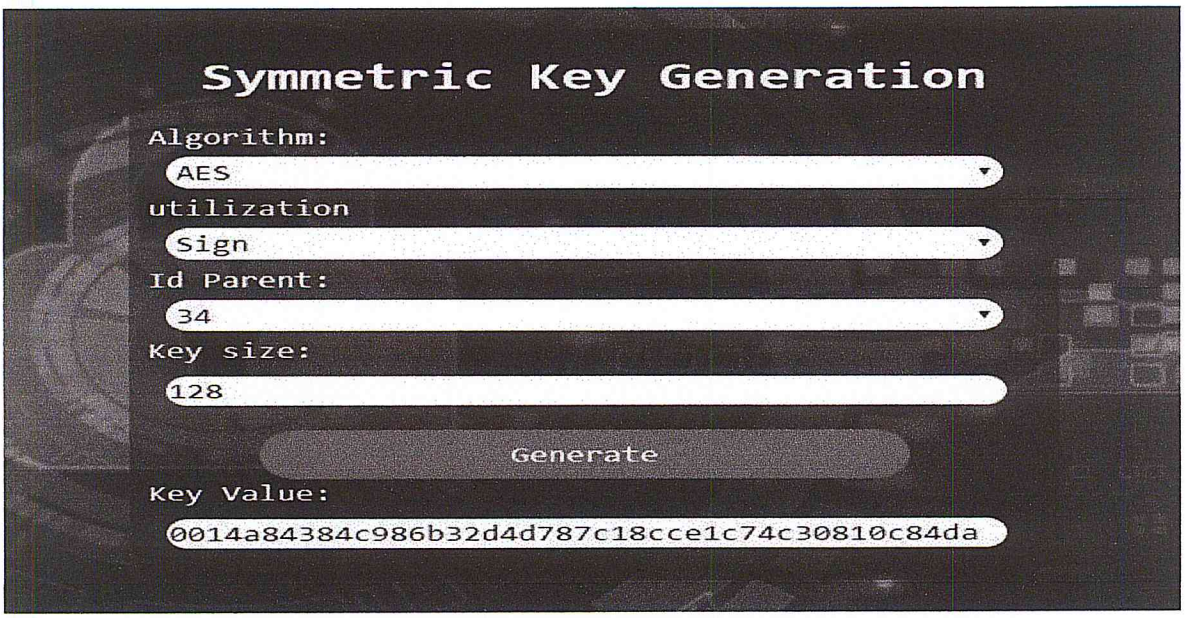


Figure 36 : la clé child AES dans l'interface user

La clé dans le Dictionnaire DCKMS :

idkey	Sym_key	idTemplate	Handle	hachkey
34	0100b62ccb3718b96feb4e355c60d3b5aabb944e0f907d0ebc...	1	-2147483648	b6745eeb2217ebe575a9df5562afb39b7a5491ad50be1d1087...
35	0100b62ccb3718b96feb4e355c60d3b5aabb944e0f907d0ebc...	1	-2147483648	5f2e84ab23ac368b4b758a9ede03a7851272d1dc31bef5c833...
36	0100b62ccb3718b96feb4e355c60d3b5aabb944e0f907d0ebc...	1	-2147483647	67221ffa32f49ac2d0cf3c1697629c10e1f4fb4866019cadca...
53	769cdb3089a52eb6560d6f1984460840fadc531ab242882f54...	1	-2147483647	899830e75873c3bd3b90662077934c698248fbe3999146fb33...
54	658ea34dc9d96a755685f34ce135feab700a319ecabd7ea666...	1	-2147483647	7225c696bd2fa20a3f90a19c1e77ab130dbb559166c83aaa90...

Ces différentes clés sont sécurisées car même le fournisseur des clés ne connait pas méthodes qu'elles ont utilisées pour la création des clés. La clé sort chiffrer d'âpres le TPM, juste l'utilisateur qui peut le déchiffrer donc rester confidentiel au cours de la route, en plus le TPM garantit l'authenticité car il utilise la signature pour vérifier si sa propre clé ou bien d'un autre TPM .En plus dans notre implémentation nous avons utilisé la session pour fermer la porte entre les différents utilisateurs de TPM car la session chiffre tous les commandes qui s'exécute à l'intérieur, ce qui garante notre solution.

V.5-Conclusion

Au fil de ce dernier chapitre, nous avons présenté la mise en œuvre de notre solution CKMS en tant que SecaaS dans le cloud, qui été la gestion des clés de chiffrement d'une façon confidentielle basé sur le TPM en tant que base de confiance et source de confiance pour la génération des clés.

Ainsi, nous avons montré l'efficacité de l'approche en exécutant différentes requêtes sur des données chiffrées sans les déchiffrer au niveau du CKMS, afin de les manipuler. Ce qui va assurer la protection des données du client

Conclusion Général et perspectives

Le marché de l'informatique en nuages (Cloud Computing) n'en est qu'à ses débuts. Mais, de l'avis de tous les experts, il dispose d'un fort potentiel de croissance, poussé notamment par la pression financière que subissent les entreprises dans un climat économique tendu. Ce marché englobe plusieurs acteurs, allant de l'éditeur de logiciels à l'expert en virtualisation en passant par l'opérateur de télécoms. Sur les différents segments de marché du Cloud, l'édition de logiciels (SaaS), avec tout un écosystème de partenaires en train de se fédérer autour d'elle, y occupe une position clé, vu la diversité de solutions qu'elle offre, notamment dans le domaine de la sécurité. Plusieurs travaux de recherche sont lancés à cet effet, entre autre la proposition de solutions de type Security as a Service (SecaaS) comme une plateforme de confiance pour la gestion des clés de chiffrement et leur sécurité, le travail présenté dans ce mémoire s'inscrit dans ce cadre.

Dans ce travail, nous avonsprésenté un état de l'art sur le Cloud-Computing, ses challenges de sécurité, la gestion de clés de chiffrement (CKMS) et le problème de confiance qui se pose, et enfin le TPM en tant que solution aux problèmes de confiance entre le fournisseur et l'utilisateur.

Par la suite nous avons présenté l'approche d'architecture de CKMS en tant que SecaaS basé sur le TPM avec un processus de création des clés. Nous avons modélise notre solution par la méthode UML. Dans la partie développement nous avons utilisé un simulateur (TPM2.0) pour l'implémentation de l'ensemble de fonctions de gestion des clés. A cet effet, nous avons réalisé des expérimentations sur un jeu de données, afin d'évaluer l'approche et de montrer qu'elle permet bien la création et le chiffrement des clés avec leurs transfert vers leurs owner d'une manière sécurisée et confidentielle.

Au but de ce travail nous avons trouvé plusieurs difficultés car nous avons utilisé le simulateur TPM2.0 qui bug presque tous les temps et bloque lorsqu'on lance plusieurs commande à la fois ce qui est limité notre travail à cause de ce problème nous avons éliminé la partie de chiffrement de la clé avec plusieurs clés.

Afin d'enrichir notre solution proposée, nous envisageons les améliorations suivantes :

- L'utilisationde la partie d'authentification on utilise le TPM2.0.

- L'établissement d'un système de sécurité basé sur le HMAC, qui va assurer l'intégrité des données transitant entre le client et le serveur, ainsi que la sécurisation de ce transit dans le contexte Cloud
- La création d'autres types de clés avec autres algorithmes dans TPM20 (ECC).
- La signature de la clé avec la clé publique de TPM

Ce projet a fait l'objet d'une expérience intéressante nous permettant à la fois d'améliorer et d'acquérir de nouvelles connaissances dans le domaine de la sécurité en général et des systèmes de gestion de clefs de chiffrement en particulier, mais aussi de découvrir de nouveaux domaines tels que le Cloud Computing, le TPM comme une technologie prometteuse d'avenir.

Bibliographie

[1] R. L. &. V. R. D. Krutz, Cloud security: A comprehensive guide to secure cloud computing, Wiley Publishing, 2010.

[2] «Cloud Computing,» [En ligne]. Available: www.missarte.wordpress.com. [Accès le 01/11/2017].

[3] L. F. NOUMSI, «Etude et mise en place d'une solution "cloud computing " privée dans une entreprise moderne: cas de CAMTEL,» Ecole nationale supérieure des postes et télécommunications - Ingénieur des travaux des télécommunications, 2012.

[4] P. Grange, Le livre blanc du cloud Computing-Tout ce que vous devez savoir sur l'informatique dans le nuage, Syntec informatique – 2ème Trimestre, 2010.

[5] B. F. A. Escalante, Handbook of Cloud Computing, Springer Science&Business Media, USA, 2010.

[6] R. V. C. &. S. S. T. Buyya, Mastering cloud computing: foundations and applications programming, Newnes, 2013.

[7] «Quadtec solutions INC,» [En ligne]. Available: www.quadtec.com. [Accès le 10/10/2017].

[8] M. Mimoune, « Etude sur la sécurité du cloud computing,» Université De Mesila , 2014.

[9] Winkler.V.J, «Securing the Cloud: Cloud computer Security techniques and tactics,» *Elsevier .*, 2011.

[10] «Le cloud computing,» [En ligne]. Available: www-igm.univ-mlv.fr/~dr/XPOSE2009/cloudcomputing/types.html. [Accès le 05/09/2017].

[11] G. Reese, Cloud Application Architectures, Edition O'Reilly Media , 2009.

[12] M. &. A. H. Hussain, «SECaaS: security as a service for cloud-based applications,» *In Proceedings of the Second Kuwait Conference on e-Services and e-Systems ,* n° %1ACM, p. p. 8, 2011, April.

[13] J. A. ,. A. L. ,. P. ,. R. ,. M. Rich Mogull, CSA Security Guidance for Critical Areas of Focus in Cloud Computing v4.0, Editors Dan Moren John Moltz , 2017.

[14] S. Ghernaouti-Helie, Sécurité informatique et réseaux - Cours et exercices corriges, 3ème édition, 2011.

[15] A. Belkhir, Cour Sécurité des Réseaux, USTHB, 2011.

[16] M. S. D. B. S. ,. Elaine Barker, A Framework for Designing Cryptographic Key Management Systems, NIST DRAFT Special Publication 800-130, April,2012.

[17] P.-L. Cayrel, «Fonction de hachage et signatures électroniques,» Université de Limoges, France, 2007. [En ligne]. Available:

http://www.cayrel.net/IMG/pdf/Fonction_de_hachage_et_signature.pdf. [Accès le 01/06/2017].

[18] G.Labouret, «Support de cours du cabinet Hervé Schauer, Introduction à la cryptographie,» 2001. [En ligne]. Available: http://www.hsc.fr/ressources/cours/crypto/crypto.pdf.. [Accès le 21/04/2017].

[19] Cathalo, «Cathalo L'importance de la taille des clés en cryptographie,,» Julien 2012. [En ligne]. Available: http://www.blogresearch.smalsrech.be/?p=4299. [Accès le 21/11 /017].

[20] «cryptomathic,» [En ligne]. Available: www.cryptomathic.com. [Accès le 14/02/2017].

[21] «cloudpatterns,» [En ligne]. Available: http://cloudpatterns.org/mechanisms/cryptographic_key_management_system. [Accès le 15/11/2017].

[22] «SCAN,» [En ligne]. Available: https://www.scan.co.uk/products/gigabyte-gc-tpm-20-trusted-platform-module-compute-securely?v=c.. [Accès le 05/04/2018].

[23] L. Duflot, F. G. ANSSI et e. t. l. n. d. d. T. C. G. AMOSSYS, Trusted Platform Module Library Part 1: Architecture . Family "2.0" Level 00 Revision 01.38, September 29, 2016.

[24] W. e. D. C. Arthur, A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security, 2015.

[25] M. Fowler, The new methodology, Wuhan University Journal of Natural Sciences, 6(1-2), 12-24, 2001.

[26] G. P. Nathalie et M. Alain , Modélisation objet avec UML, Eyrolles edition, Avril 2005.

[27] [En ligne]. Available: www. apachefriends.org. [Accès le 06/06/2018].

[28] «Microsoft,» [En ligne]. Available: www.microsoft.com. [Accès le 15/03/2018].

[29] [En ligne]. Available: www.apachefriends.org. [Accès le 11/06/2018].

[30] «Conception-et-realisation-dun-site-web,» [En ligne]. Available: www.memoireonline.com. [Accès le 11/06/2018].

[31] [En ligne]. Available: openclassrooms.com. [Accès le 11/06/2018].