UNIVERSITE SIDI MOHAMED BEN ABDELLAH
FACULTÉ DES SCIENCES ET TECHNIQUES FÈS
DÉPARTEMENT D'INFORMATIQUE

PROJET DE FIN D'ETUDES

MASTER SCIENCES ET TECHNIQUES
SYSTÈMES INTELLIGENTS & RÉSEAUX

LA SÉCURITÉ DANS LE CLOUD COMPUTING

LIEU DE STAGE : LSIA

RÉALISÉ PAR : EL GHOUBACH IMAD

SOUTENU LE 18 JUIN 2014

ENCADRÉ PAR :

MR PR.R. BENABBOU
MME PR.F.MRABTI

DEVANT LE JURY COMPOSÉ DE :

MR PR.R. BENABBOU
MME PR.F.MRABTI
MR PR. A.ZAHI
MME PR. CHAKER ILHAM

ANNÉE UNIVERSITAIRE 2013-2014

Remerciement

Après Dieu, je tiens à remercier :

Monsieur BENABBOU Rachid, sans qui l'avancement de ce projet n'aurait pas été possible, avec ses qualités de conseils et son orientation pour l'élaboration de ce travail.

Professeur MRABTI Fatiha, pour l'aide qu'elle m'a apporté, ses précieuses recommandations ainsi son encouragement qui m'a été adressé durant tout le déroulement de ce travail.

J'aimerais également remercier tous les enseignants du département informatique de la Faculté des Sciences et Techniques de Fès pour le fait d'avoir contribué à notre formation en Master SIR et de nous avoir permis de s'enrichir dans le domaine de l'informatique.

Je remercie ainsi mes parents pour leur soutien et leurs conseils durant mes études, mes amis pour leurs encouragements et leur soutien et toute personne qui a contribué de près ou de loin à l'avancement de ce travail.

Résumé

Cloud Computing a été envisagé comme l'architecture de la prochaine génération des technologies de l'information. Contrairement aux solutions traditionnelles, où le contrôle (physique et logique) et la gestion des services informatiques sont sous la supervision du propriétaire des données. Le Cloud Computing externalise les applications et les bases de données vers les centres de données de fournisseur de service, où la gestion des données et des services peut ne pas être entièrement digne de confiance, soit à cause de l'immaturité de la sécurité de ce domaine, soit par manque de confiance dans le fournisseur de service lui-même. Dans le cadre de notre projet, nous avons établis une étude bibliographique de la sécurité du Cloud Computing en se focalisant principalement sur la sécurité des données. Dans cette étude, nous avons effectué une analyse comparative des différentes solutions abordées dans la sécurité des données.

Abstract

Cloud Computing has been envisioned as the architecture of the next generation of IT. Unlike traditional solutions, where IT services (physical, logical) are controlled and managed under the supervision of the data owner. Cloud Computing outsources applications and databases to the data center service provider, where the management of data and services may not be fully trusted , due to immaturity of the security in this technology or to the lack of trust in the service provider itself. In our project, we have established a bibliographic study of Cloud Computing focusing primarily on data security. In this study we performed a comparative analysis of the different solutions addressed in data security.

Sommaire

Liste des Figures

Liste des tableaux

List des Acronymes

API
Application Programming Interface 26, 37
ASP
Application service provider 18
BEL
Basic Encryptions Layer 69
DES
Data Encryption Standard 50
DMA
Direct Memory Access 7, 40, 41, 42, 44, 59
DOS
Deny of service 40
IaaS
Infrastructure comme service 4, 5, 12, 24, 27, 36, 37, 93
KP-KBE
Key Policy Attribut based Encryptions 62
LSIA
Laboratoire Systèmes Intelligents et Applications 12
MAC
Message Authentication Code 6, 7, 52, 53, 76, 80
MK
Master Key 63
NIST
National Institute of Standards and Technology 15
PaaS
Plateforme as a service 12
PK
Clé public 63, 64
SaaS
Software as a service 12
SEL
Surface Encryptions Layer 70
SK
Secret Key 65, 66
TCB
Trusted Computing base 6, 56
UL
User List 66
VM
Virtual Machine 20

Introduction

1. Introduction du laboratoire

Ce travail a été réalisé dans le cadre d'un stage au Laboratoire Systèmes Intelligents et Applications (LSIA), crée en 2011. C'est une unité de Recherche du Centre d'Etudes Doctorales en Sciences et Techniques de l'Ingénieur domicilié à la Faculté des Sciences et Techniques de Fès et regroupant 17 laboratoires de recherche tous accrédités par l'Université Sidi Mohamed Ben Abdellah de Fès, et domiciliés à la Facultés des Sciences et Techniques, l'Ecole Supérieure de Technologie et la Faculté Polydisciplinaire de Taza.

Le LSIA est composé de 13 enseignants-chercheurs du département d'Informatique de la FST de Fès et de 8 doctorants. Cette imbrication étroite entre enseignement et recherche, est un élément essentiel de la dynamique du laboratoire.

Les thématiques de recherche se situent au cœur des Sciences et Technologies de l'Information et de la Communication et s'articulent essentiellement autour des thématiques de recherche des enseignants chercheurs du laboratoire et assure une large couverture thématique présentant un atout très important pour le LSIA.

Le LSIA est organisé en trois équipes :

- Systèmes de Communication et Traitement de Connaissances (SCTC),
- Environnement Intelligents & Applications (VIA),
- Vision Artificielle & Systèmes Embarqués (VASE).

2. Introduction du projet

Le Cloud Computing est un paradigme émergeant dans l'industrie de la technologie d'information. Cette technologie a comme but d'externaliser les calculs et les ressources vers un nuage[1] d'ordinateurs, cela est possible grâce au vaste avancement dans les technologies de virtualisation, web et la bande passante ce qui a permis d'offrir des ressources comme services. Ces dernières sont généralement servies sous trois catégories :

- Software comme service (SaaS) : cette catégorie vise à offrir des applications offrant des services bien précis comme Google docs. Cette application peut être développée soit par le fournisseur de service, soit par un développeur tiers dans la couche PaaS.

- Plateforme comme service (PaaS) : dans cette couche les fournisseurs visant à offrir des environnements de développement permettant aux startup et aux programmeurs de développer et de déployer les applications en offrant des APIs et des outils offrant un accès aux ressources à travers une couche de virtualisation.

- Infrastructure comme service (IaaS) : cette couche offre la possibilité de réserver des ressources matérielles au du besoin en payant seulement une portion du coût des ressources matérielles réelles.

[1]Nuage: est un terme utilisé comme une métaphore pour les réseaux étendus (comme Internet) ou d'un tel grand environnement réseau.

Cette technologie a permis aux organisations, spécialement les startups de réduire le coût des investissements et de débarrasser des coûts de gestion et de la maintenance, ce qui a permis au le Cloud à se développer très rapidement. En effet en 2012, le Cloud a pris 9% (42 billions de dollars) de la totalité des investissements dans le domaine de la technologie d'information.

Pourtant, malgré les avantages économiques et techniques offerts, l'activité du Cloud n'a pas encore atteint sa maturité. En effet en 2009, IDC [2] a annoncé que la majorité des entreprises n'ont pas encore envisagé la migration au Cloud à cause des problèmes reliés à la confiance et à la sécurité de leur données.

L'utilisation du Cloud signifie que les clients doivent externaliser leurs données en les mettant chez un fournisseur de service ce qui impose la nécessité de prendre des précautions pour protéger la confidentialité de ses données. De plus ils doivent avoir suffisamment de preuves que les informations stockées chez le fournisseur ne sont pas modifiées et sont accessibles à la demande seulement par le propriétaire ou quelqu'un qui a les droits d'accès. Cela relève des problèmes d'intégrité, de disponibilité et de confidentialité.

En effet le Cloud est né d'une collaboration des technologies web, virtualisations et de la bande passante ce qui signifie qu'il n'hérite pas seulement leurs points forts, mais aussi leurs faiblesses. A vrai dire, grâce à la centralisation des ressources, l'impact de ces failles augmente de manière exponentielle, pour cette raison des études ont été lancées pour remédier à ces lacunes.

Dans le cadre de notre travail, nous avons établi une étude bibliographique sur les problèmes de sécurité et les solutions proposées. Dans le premiers chapitre nous avons commencé par introduire le Cloud, sa définition, son historique et ses modèles, puis nous avons essayé dans la deuxième chapitre d'illustrer les problèmes de sécurité dans chaque niveau du modèle de service en se focalisant sur les problèmes de sécurité dans la virtualisation et la sécurité des données comme étant la partie vitale de la sécurisation du Cloud. Enfin, dans la troisième chapitre nous avons présenté les différentes solutions existantes dans la virtualisation ainsi qu'une comparaison entre les solutions proposées dans la confidentialité et l'intégrité des données dans le Cloud Computing.

Chapitre I : Cloud Computing

3. Introduction

Le Cloud Computing se traduit littéralement par "informatique dans les nuages", faisant référence aux technologies d'Internet, il est souvent représenté schématiquement par un nuage, c'est un concept abstrait qui regroupe plusieurs technologies servant à délivrer des services pour permettre aux entreprises d'externaliser leurs ressources numériques. Ces ressources, offrant des capacités de stockage et de calcul, des logiciels de gestion de messagerie et d'autres services, sont mises à disposition par des sociétés tierces et accessibles, grâce à un système d'identification et une connexion Internet.

Ce chapitre se compose de quatre parties principales. Premièrement nous allons commencer par citer les différentes définitions du Cloud existantes dans la littérature, suivit d'un positionnement de Cloud Computing dans l'histoire de la technologie de l'information, , ensuite nous allons introduire la virtualisation comme étant l'une des principaux piliers de cette technologie. Enfin nous présenterons les deux modèles du Cloud Computing, à savoir le modèle de service et le modèle de déploiement.

4. Définitions

Le Cloud Computing est un paradigme défini de plusieurs manières par plusieurs organisations. En effet, il est difficile de lui donner une définition exacte, de nombreux organisations utilisent le terme Cloud à des fins marketings, ce qui impose la question « qu'est-ce que le Cloud Computing ? »

Ce concept a été défini de plusieurs manières, telles que:

- Wikipédia définit le Cloud Computing comme étant un concept qui consiste à déporter sur des serveurs distants des traitements informatiques traditionnellement localisés sur le poste client de l'utilisateur**.**
- Pour NIST [3] le Cloud Computing est une nouvelle façon de délivrer les ressources informatiques et non une nouvelle technologie" **.**
- Syntec [4] définit le Cloud Computing comme une interconnexion et une coopération de ressources informatiques, situées au sein d'une même entité ou dans diverses structures internes, externes ou mixtes et dont le mode d'accès est basé sur les protocoles et standards Internet.

En effet le Cloud Computing peut être vu comme un système d'exploitation distribué sur des milliers de machines assurant l'abstraction de l'infrastructure et permettant d'héberger et d'exécuter des applications et de fournir des services (stockage, calcul, traitement de données etc.).

Généralement le Cloud Computing se compose de cinq caractéristiques essentielles [3] :

- **Accès « à la demande » et « self-service » par l'utilisateur**

  L'utilisateur peut accéder à la ressource au besoin sans passer forcément par une interaction humaine avec le fournisseur de service.

- **Accès Internet large bande**

  Les services fournis sont accessibles via Internet, en utilisant des mécanismes permettant l'utilisation de ces services, par des plateformes hétérogènes (smartphones, tablettes, pc portables ...).

- **Multitenance (ressources partagées)**

  Les fournisseurs de services Cloud fournissent des capacités de ressources partagées par tous les clients. Ces capacités sont attribuées et réattribuées selon la demande et le besoin de l'utilisateur. En effet, le client n'a aucun contrôle sur la localisation de ces données et des serveurs faisant le traitement des données.

- **Redimensionnement rapide (élasticité)**

  L'utilisateur peut facilement changer la taille de la ressource utilisée (augmenter ou diminuer) selon le besoin, comme il peut la libérer à la fin de l'utilisation.

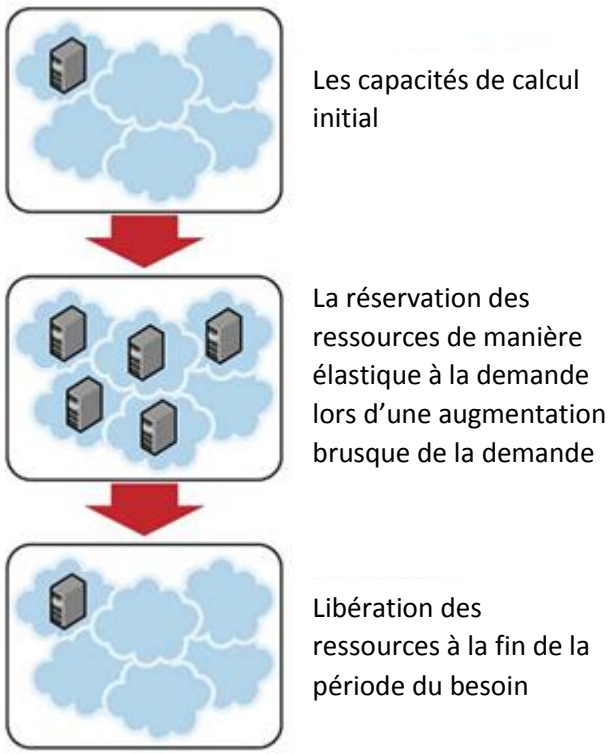


Figure I-1 Elasticité

- **Services mesurés**

  Le fournisseur de service peut facilement contrôler et optimiser les ressources utilisées en utilisant des capacités métriques dans un certain niveau d'abstraction approprié au service, cela permet le suivi, le contrôle et le rapportage des ressources offrant une transparence pour les fournisseurs comme pour l'utilisateur. Ce concept a permis l'utilisation du concept de facturation qui permet à l'utilisateur de payer seulement les ressources utilisées.

Les différentes caractéristiques du Cloud Computing ont vu le jour grâce aux progrès important dans le domaine de la virtualisation qui a permis d'optimiser les ressources matérielles en les partageant entre plusieurs environnements « time-sharing ». En plus La virtualisation a donné la possibilité de lancer plusieurs instances d'applications ou des systèmes d'exploitation et la virtualisation du matériel (sous forme de machines virtuelles), cela a permis d'assurer la capacité de déploiement d'environnement, de manière automatique.

5. Historique

La notion du Cloud Computing n'a pas une date de naissance précise, mais le concept n'est pas nouveau. Le premier concept similaire au Cloud a été présenté en 1961 par John McCarthy qui a proposé, dans un discours, une vision dans laquelle les puissances de calcul et même des applications spécifiques sont vendues comme services publics, cette idée était populaire dans les années 60 mais elle a disparu dans les années 70 à cause du manque des technologies nécessaires.

Une autre technologie appelée virtualisation a été développée dans les années 60 dans les centres de recherche de Cambridge et de Grenoble. La virtualisation considérée comme la première pierre vers le Cloud Computing permet de lancer plusieurs instances de système d'exploitations ou d'applications sur le même hardware ce qui a permis de partager les ressources (la multitenance).

Dans les années 70, la notion des services bureau a connu vie pour qualifier les entreprises louant des services comme les lignes téléphoniques, les répondeurs automatiques, les services informatiques etc. Généralement, les clients de ce type de services n'avaient pas l'expertise pour employer ces services en interne, c'est pourquoi ils passent par un prestataire. La combinaison de technologies, processus et expertise dans le domaine des entreprises est la valeur ajoutée des « services bureau », comme modèle économique basé sur leur capacité à produire des services et à les déployer en volume. IBM lui-même était un « service bureau », en proposant la notion de « on-demande » mais le coût d'achat et d'emploi des mainframes IBM était très élevé, c'est pour cette raison que des solutions permettant d'exploiter ce genre de solution au moindre coût « payer à la consommation » et au moindre prix ont été proposées.

Après l'apparition de la notion de services bureau, une autre notion a été présentée, il s'agit de l'ASP[2] (Application Service Provider), qui désigne l'externalisation des applications

[2] ASP (Application service provider) : une entreprise qui fournit des logiciels ou des services informatiques à ses clients au travers d'un réseau (Internet en général).

vers un fournisseur d'application pour permettre à l'organisation de se focaliser sur les métiers premiers. Cette notion est similaire au SaaS (Software As a Service) dans le Cloud actuel. Plutôt que d'installer le logiciel sur le poste client en ayant à assurer les phases d'installations et de maintenance sur chaque poste, les applications ASP sont hébergées et centralisées sur un serveur unique et accessible par les clients à travers des protocoles standards, dans ce cas, l'organisme utilise l'application sans se soucier du déploiement, de configuration, d'hébergement ou de maintenance de l'application, qui sont désormais de la responsabilité du fournisseur du service.

C'est donc depuis presque 50 ans que l'idée d'une informatique à la demande est présente dans les esprits, mais l'implémentation de cette idée n'était possible qu'avec les différents progrès technologiques réalisés depuis, tant sur le plan matériel, logiciel et conceptuel, à l'élaboration de réseaux complexes mais standardisés comme Internet, et à l'expérience dans l'édition et la gestion de logiciels, services, infrastructures et stockage de données. Nous sommes maintenant prêts à entrer dans l'ère du Cloud Computing, telle que rêvait John McCarthy en 1961.

6. La virtualisation

Le terme virtualisation a été inventé dans les années 60 pour faire référence au Virtual Machines. La création et la gestion de cette virtualisation ont été référencées par le terme virtualisation de la plateforme qui illustre l'idée de créer un ou plusieurs environnements pour les programmes des clients.

La virtualisation ne se limite pas seulement à la création des environnements mais peut être aussi une virtualisation des systèmes d'exploitation tout entier, cette virtualisation impose des pénalités sur les ressources, ce qui impose l'utilisation des hyperviseurs.

6.1. Hyperviseur

Un hyperviseur est la partie logicielle ou matérielle responsable du fonctionnement des machines virtuelles, il présente une plateforme d'exploitation des systèmes d'exploitation. Il existe deux types d'hyperviseurs [5] :

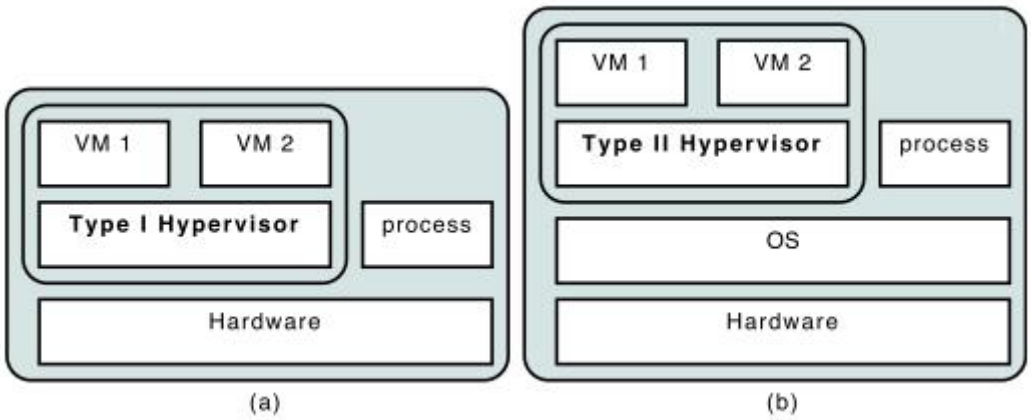


Figure I-2: Les types d'hyperviseurs [6]

➔ Type 1 (natif, Paravirtualisation ou « baremetal ») : ce modèle représente l'implémentation classique des machines virtuelles, dans ce type l'hyperviseur s'exécute directement au-dessus du matériel et gère les systèmes des clients, ce qui fait que ces systèmes s'exécutent à un niveau au-dessus de l'hyperviseur (exemple : Xen, Oracle VM, ESX Server de VMware, TRANGO).

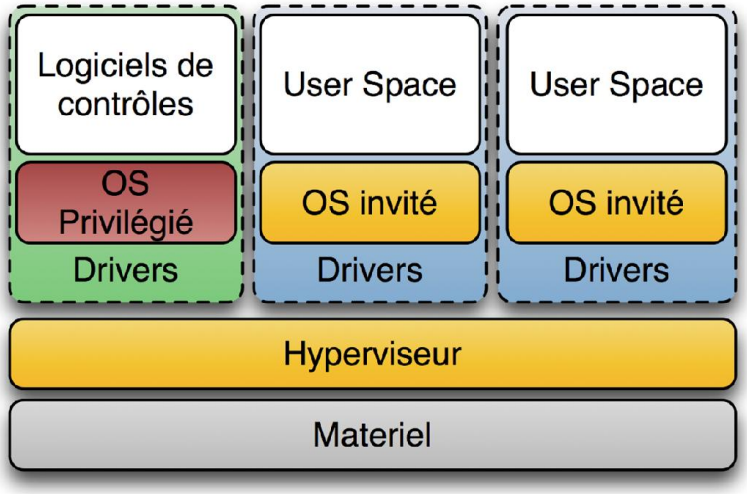


Figure I-3 : Hyperviseur natif (type 1)

➔ Type 2 (hébergé) : ce type s'exécute dans les systèmes d'exploitation, ce qui fait que les systèmes virtuels s'exécutent dans le troisième niveau au-dessus du matériel avec l'hyperviseur comme le deuxième niveau (exemple Virtual PC et Virtual Server, Virtual Box).

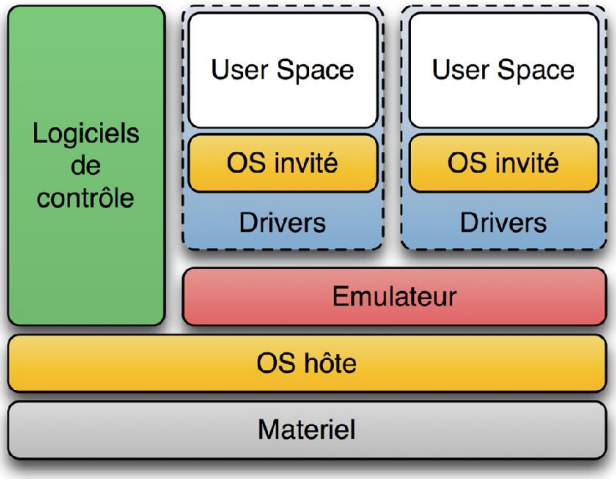


Figure I-4 : Hyperviseur hébergé (type 2)

6.2. Types de virtualisation

La propagation du virtualisation dans le domaine IT a donné naissance à plusieurs architectures qui implémentent ce concept, dans ce qui suit nous allons traiter les architectures les plus communes[6-8] :

6.2.1. virtualisation totale

Cette technologie offre la possibilité de virtualiser la totalité du matériel ce qui fait que chaque entité peut être exécutée de la même manière avec laquelle elle s'exécute dans le vrai matériel. Comme expliqué dans la figure I-5, chaque VM a l'accès à son matériel virtuel et doit demander à l'hyperviseur l'autorisation pour avoir l'accès au vrai matériel.

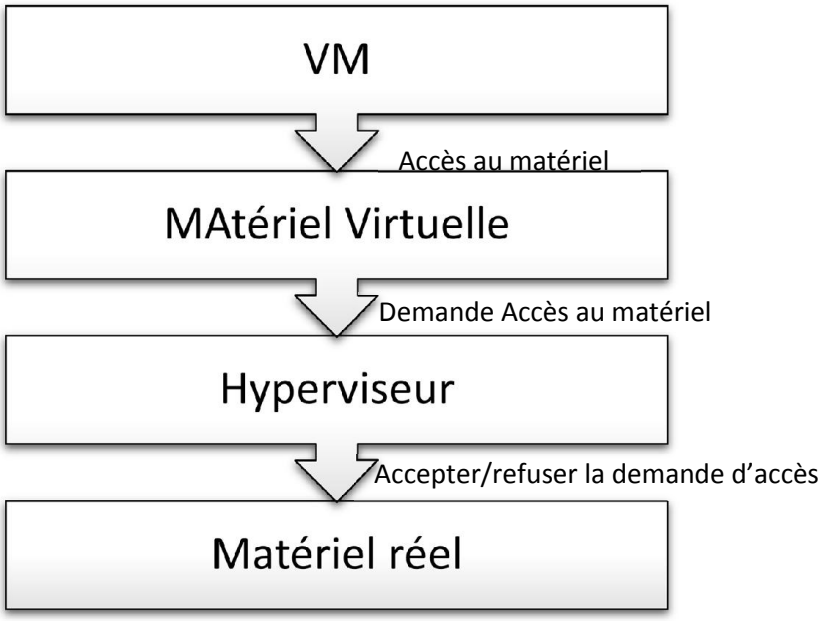


Figure I-5 Virtualisation totale

Cette virtualisation se décompose en deux familles[6] :

6.2.1.1. Software-assisted virtualisation

Cette famille représente la première approche présentée par IBM en 1969 dans laquelle tout est simulé, ce qui veut dire que chaque système d'exploitation et application peut s'exécuter sans modification, cette méthode a un coût très élevé au niveau de ressource, car chaque opération doit être simulée et vérifiée pour qu'elle n'interfère pas avec les autres systèmes et avec l'hyperviseur.

6.2.1.2. Hardware-assisted virtualisation

Différemment à la famille précédente, cette virtualisation permet d'exécuter des systèmes d'exploitation et des applications non modifiés, cette méthode étend le hardware avec de nouvelles instructions qui ont introduit des nouvelles fonctionnalités permettant d'accéder directement au hardware.

6.2.2. *Para-virtualisation*

Dans ce type de virtualisation, les machines virtuelles doivent être modifiées pour introduire la notion de l'Hypercall [6, 7, 9], chaque appel système doit être remplacé par le code source de l'application/système pour être virtualisé en appelant l'hyperviseur (Hypercall), ce qui donne la possibilité de faire des appels systèmes à l'hyperviseur sans avoir besoin de faire des simulations matérielles.

Comme illustré dans la figure I-6, dans plusieurs cas la machine virtuelle envoie les demandes directement à l'hyperviseur pour permettre une performance plus élevée, ce type de virtualisation est largement utilisé par Xen et « Parallels Workstation» [6].

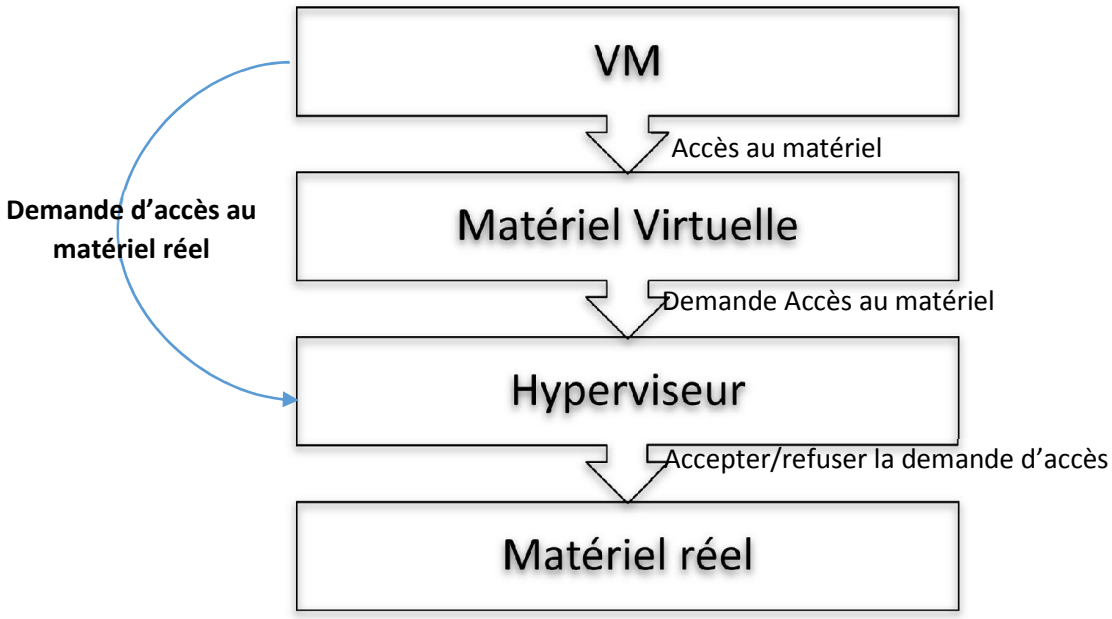


Figure I-6: Paravirtualisation

6.2.3. *La virtualisation partielle*

Dans ce type de virtualisation, la machine virtuelle simule seulement une partie du matériel. L'entité virtualisée a besoin donc de quelques modifications, si elle veut utiliser une partie du matériel non simulé[6, 8]. Souvent, il est impossible d'exécuter un système d'exploitation sous ce type de machines virtuelles. Les implémentations les plus connues de ce type de virtualisation est la virtualisation de l'espace d'adressage, où chaque processus possède un espace d'adressage complet sans avoir besoin de la quantité totale d'espace mémoire sur le matériel.

6.2.4. La virtualisation au niveau du système d'exploitation

Comme illustré dans la figure I-7 le système d'exploitation joue le rôle d'un hyperviseur partagé entre les machines virtuelles isolées[8, 9], ce type de virtualisation permet de partager les ressources matérielles entre les différentes machines sans avoir de risques d'interférence, ce qui encourage l'apparition de plusieurs implémentations comme OpenVZ et Linux-VServer. Le problème de ce type de virtualisation est que les machines virtuelles et la machine hôte doivent utiliser le même système d'exploitation.

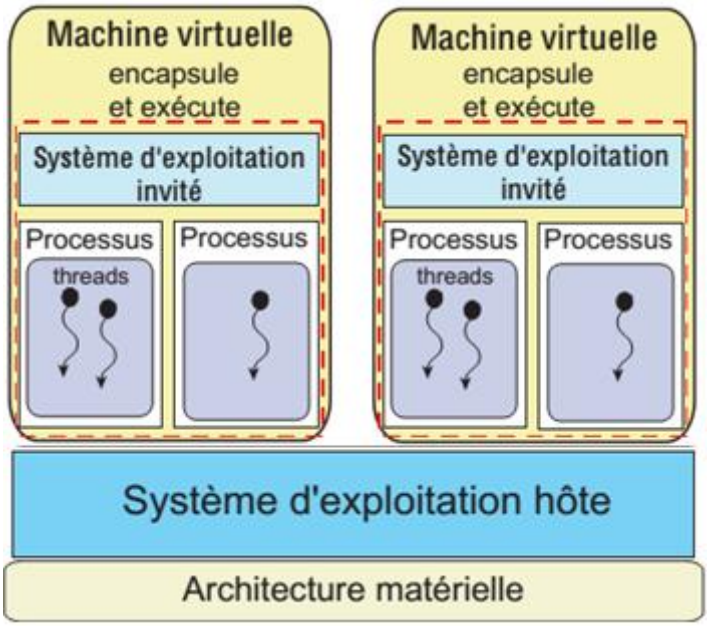


Figure I-7: La virtualisation au niveau du système d'exploitation

6.2.5. La virtualisation du matériel

Le protocole de partage des ressources établi par l'hyperviseur peut être séparé en deux catégories[6, 9] :

➔ Hyperviseurs à isolation pure : comme illustré dans la figure I-8, chaque client a son propre matériel et drivers dédiés, non partagé avec les autres clients, ce qui fait que cette méthode est la plus simple et la plus sécurisée.

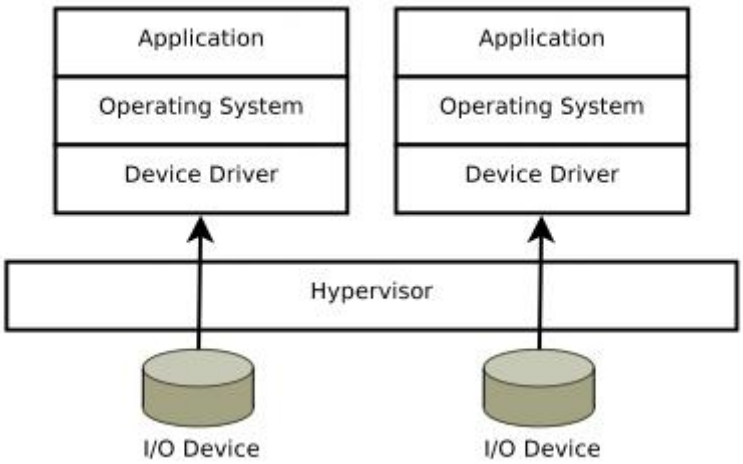


Figure I-8 l'hyperviseur à isolation pure[6]

➔ « Sharing » hyperviseur : avec cette méthode le matériel et les drivers sont partagés, il existe deux méthodes principales pour cela :

- Avec la première méthode, les drivers sont intégrés dans l'hyperviseur et ont l'accès seulement au matériel.

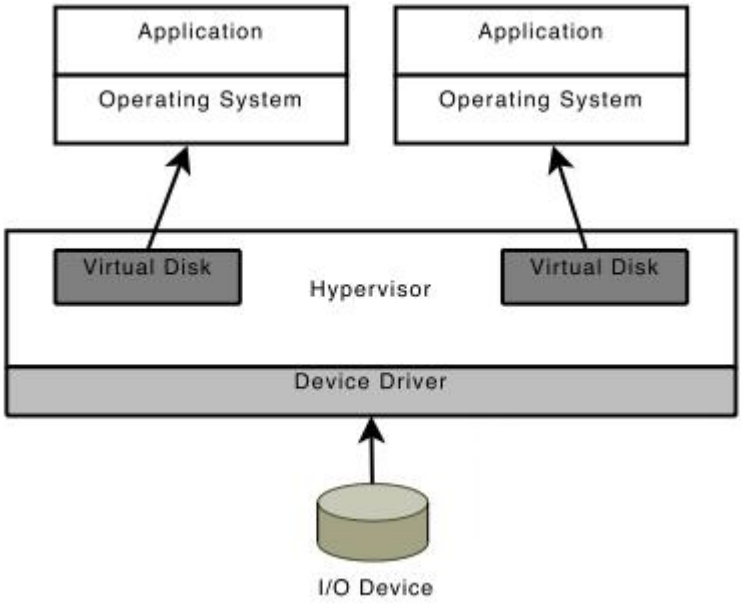


Figure I-9 : La première méthode de partage[6]

- Dans la deuxième méthode, une machine privilégiée est créée pour faire la gestion de tous les matériels partagés pour les autres clients via VMM, les drivers fonctionnent seulement dans ce client privilégié, ce principe est implémenté dans le cas de Xen, mais comme nous verrons par suite, avec cette implémentation, de nouveaux risques de sécurité doivent être pris en considération.

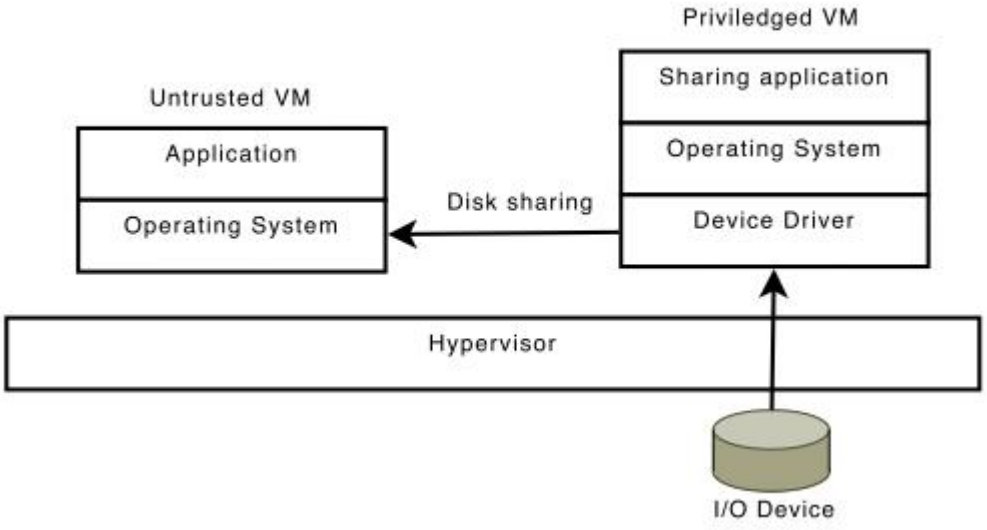


Figure I-10: La deuxième méthode de partage[6]

Le Cloud Computing couplé, aux technologies de virtualisation, permet la mise à disposition d'infrastructures et de plateformes à la demande. Mais le Cloud Computing ne concerne pas seulement l'infrastructure (IaaS), mais aussi la plate-forme d'exécution (PaaS) et les applications (SaaS).

7. Les modèles de services

Le Cloud Computing est aussi décrit par l'acronyme "SPI", désignant ainsi les trois services majeurs proposés par le Cloud : Software-as-a-Service(SaaS), Platform as-a-Service (PaaS) et Infrastructure-as-a-Service (IaaS)[10].

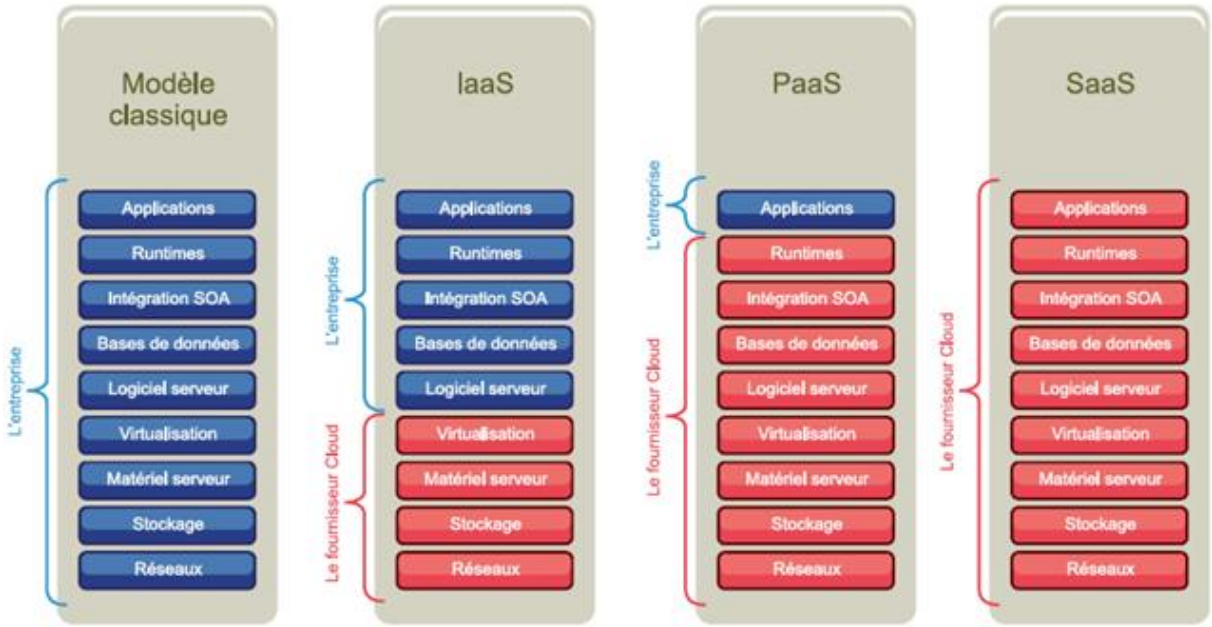


Figure I-11: Modèles de services[4]

L'acronyme Cloud Computing est né à partir d'une collaboration entre plusieurs technologies. Ces dernières ont été développées dans des contextes différents pour des buts différents.

La coopération entre ces technologies n'était pas prise en considération, mais ce développement a construit un milieu technique pour le Cloud, en plus cette technologie est devenue de plus en plus irrésistible avec le développement dans les processeurs, les technologies de virtualisation, les technologies de stockages, la connexion d'internet à haut débit et les serveurs rapides à coût minime.

Généralement le modèle de service du Cloud est composé de trois catégories [10-13]:

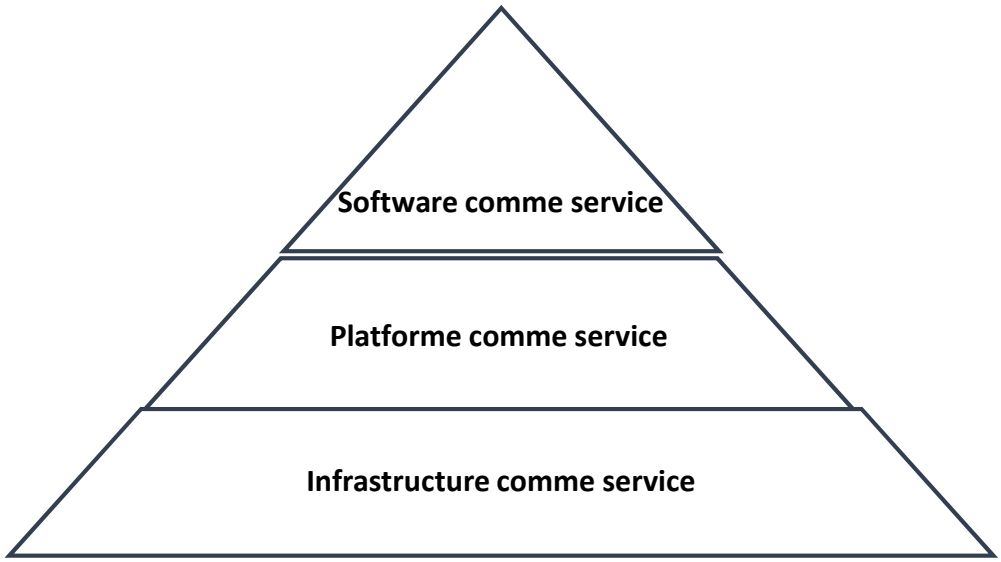


Figure I-12: Modèle de service

7.1. Software comme service (SaaS)

Ce modèle a été présenté comme alternative du modèle traditionnel, dans lequel l'utilisateur achète un logiciel avec une licence de maintenance, pour recevoir les mises à jour et les patches et reste toujours concerné par la compatibilité de ces derniers et par sa conformité avec la licence. Dans ce modèle, le client loue le service pour la période dans laquelle il va l'utiliser (pay as you go). Généralement le service loué est doté d'une indépendance complète en ce qui concerne les côtés matériel, logiciel et maintenance.

Un utilisateur d'un service dans ce modèle peut se connecter facilement à partir d'un appareil autorisé. Dans des cas, des préparations sont requises pour accéder au service, comme dans le cas des entreprises où la connexion au service peut être intégrée à un logiciel indépendant de ce modèle.

L'implémentation du SaaS permet de bénéficier de plusieurs avantages :

- Il offre aux entreprises la possibilité d'utiliser des serveurs de données externes comme moyen de réduire le coût matériel, logiciel et le coût de personnel.

- Ce modèle permet au vendeur des applications de maintenir l'usage limité de leur software en limitant le copiage et la distribution, généralement un flux de revenu est établi entre les utilisateurs et le fournisseur sans avoir besoin de pré-charger le logiciel dans chaque matériel dans une organisation.

- L'utilisateur des services du modèle SaaS n'a pas besoin de prérequis matériel, un utilisateur peut accéder à un service à partir de son navigateur en utilisant la connexion internet existante.

- Le service est géré par le fournisseur, même si l'utilisateur peut le configurer en utilisant une API (Application Programming Interface), la configuration reste limitée (un service ne peut pas être complètement modifié).

La différence essentielle entre le SaaS et le modèle traditionnel est la nature des applications ; dans le modèle traditionnel une application achetée par un utilisateur et installée dans un serveur est accessible seulement par cet utilisateur et son groupe (Application single-tenant), contrairement au SaaS qui offre des services accessibles par plusieurs utilisateurs appartenant à plusieurs groupes différents tout en conservant la confidentialité des données de chaque groupe (Application multi-tenant).

Il existe une grande différence entre les solutions SaaS et les ASP (Application service providers) :

- Les applications offertes par les ASP sont des applications qui appartiennent au modèle traditionnel (single-tenant), qui sont hébergées sur un serveur tierce en utilisant des interfaces *html* pour garantir l'accès à distance à l'application, de plus ces applications ne sont pas modélisées pour être net-natives, ce qui fait qu'elles sont d'une performance basse et d'une gestion des mises à jours et de compatibilité similaires à une application sur un serveur appartenant à l'organisation cliente.

- Les solutions SaaS sont des solutions net-natives gérées par des spécialistes, ce qui fait que le fournisseur de ces services peut garantir une excellente maintenance et gestion de ses applications.

7.2. Plateforme comme service (PaaS)

Comme son nom l'indique, dans ce modèle le fournisseur offre un environnement de développement au client qui s'en sert pour réaliser des applications pour offrir des services fournis à partir de la plateforme du fournisseur. Généralement le fournisseur développe des outils, des standards de développement et des chaînes de distribution et de payement, ainsi reçoit des payements pour la plateforme et les chaînes offertes, ce qui facilite la propagation des applications. Tout cela fait du PaaS une variété du SaaS dans laquelle l'environnement de développement est offert comme service au développeur.

La force du PaaS réside du fait qu'il permet aux développeurs et aux start-ups de déployer leur application web avec les moindres coûts et complexité, de plus, elle offre la possibilité au développeur de développer une application web, sans avoir besoin d'un niveau de connaissance approfondi et spécialisé. Cette plateforme facilite le développement des

applications du SaaS, car elle inclut des outils de développement et de déploiement multi-tenant, en plus d'une gestion et facturation intégrées.

7.3. Infrastructure comme service (IaaS)

Comme nous avons déjà mentionné, le Cloud offre la notion de « payer ce que tu consommes », ce qui fait que dans ce modèle, un fournisseur peut facilement réserver et échelonner des ressources matérielles quand il en a besoin et permet à l'utilisateur d'accéder à la meilleure ressource technologique avec un coût très confortable.

Le tableau suivant illustre des exemples de services fournis dans chaque niveau de modèle de service avec leurs avantages :

Tableau I-1 : Exemple des avantages par rapport aux services

Modèle de service	Service	Exemple	Fournisseur de service	Avantages
Software comme service (SaaS)	Le software est fourni comme service et consommé à travers le navigateur (application web)	· Excel · Access · SQL Server · CRM[3] · ERP…	· GoogleApps · Salesforce.com	· Réduire le coût · Contrôle centralisé
Plateforme comme service (PaaS)	Une plateforme est fournie au client qui lui Permet de développer et suivre l'application à travers tout le cycle de développement sans installer aucun outil	· Codage · Scripting · Intégration · Test · Déploiement	· AppEngine · Azure · Engine Yard · Force.com	· Extensibilité · Fiabilité et sécurité · Payer à l'utilisation
Infrastructure comme service	Infrastructure est louée au client	· Extensibilité et la disponibilité des infrastructures	· Amazon · EC2 · S3 · GoGrid · Linode · Rackspace	· Extensibilité · Payer ce qu'on consomme · Les meilleures technologies et ressources

[3]Customer Relationship Management : Ce sont des progiciels qui permettent de traiter directement avec le client, que ce soit au niveau de la vente, du marketing ou du service et que l'on regroupe souvent sous le terme de "front-office", ceci par opposition aux outils de "back-office" que sont les progiciels de gestion intégrés.

8. Modèles de déploiements

On peut aussi classifier le Cloud selon son modèle de déploiement ou son modèle de service [10]. Dans ce contexte, le NIST [3]distingue trois modèles de déploiement pour le Cloud :

8.1. Cloud public

Le Cloud public représente le Cloud dans la version traditionnelle du terme, c'est-à-dire un fournisseur tiers offre des services facturés à l'utilisation. Dans ce modèle ce fournisseur est responsable de la sécurité, ce qui donne aux utilisateurs un faible degré d'accès et de surveillance sur les deux aspects physique et logique du Cloud.

Figure I-13 : Cloud public

8.2. Cloud privé

Contrairement au Cloud public, le Cloud privé n'est accessible que par l'organisation pour laquelle il est dédié, cela veut dire que tous les infrastructures, réseaux, applications et stockage associés à ce modèle sont réservés à l'organisation. Plusieurs modèles sont dérivés du Cloud privé tels que :

- Cloud dédié : dans ce modèle l'organisation est propriétaire de l'infrastructure et c'est elle qui s'occupe de sa gestion.

- Cloud géré : dans ce modèle l'organisation est propriétaire de l'infrastructure mais la gestion est faite par le fournisseur.

Dans ce type de Cloud le client a un niveau de contrôle élevé de surveillance sur la sécurité des données.

8.3. Cloud Communautaire

Dans ce modèle de Cloud, l'infrastructure est provisionnée pour une utilisation exclusive par une communauté spécifique d'organisations qui ont des préoccupations partagées, elle peut appartenir, être gérée et exploitée par une ou plusieurs organismes de la communauté, un tiers, ou une combinaison d'entre eux et il peut même exister dans ou en dehors des locaux de l'organisme.

Figure I-14 : Cloud Communautaire

8.4. Cloud hybride

Une organisation peut adopter un déploiement du genre Cloud hybride, qui est un environnement qui consiste en multiples Clouds avec des fournisseurs internes ou externes. Avec cette architecture, une organisation peut déployer des applications sur un Cloud public,

tout en conservant la confidentialité de ses données et ses applications dans le Cloud privé.

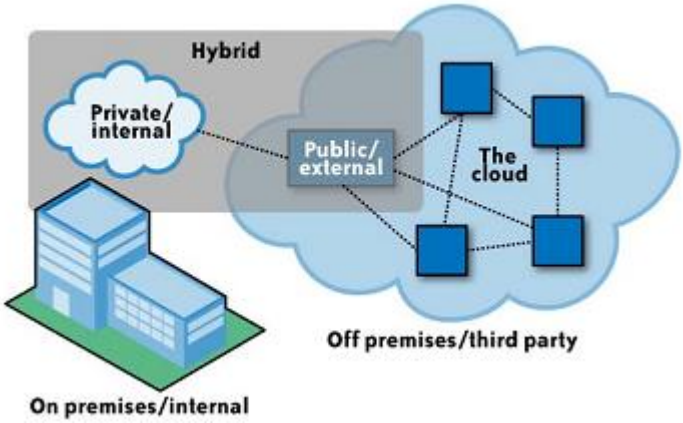


Figure I-15 : Cloud Hybride

9. Les avantages du Cloud Computing

La technologie du Cloud Computing a connu une évolution très rapide à partir de sa première implémentation par Saleforces en 1999 jusqu'à aujourd'hui. Cette évolution est due à deux raisons : la première est l'évolution technologique du matériel et de la bande passante, qui a transformé l'internet en une zone fertile pour l'évolution du Cloud, la deuxième raison est le nombre d'avantages que cette technologie offre à ses utilisateurs.

Dans le tableau ci-dessous, nous allons illustrer une comparaison entre le modèle traditionnel et le Cloud Computing :

Tableau I-2: Une comparaison entre la méthode traditionnelle et le Cloud Computing

Modèle traditionnel / dédié	Cloud Computing
Un investissement coûteux à l'avance	Un faible investissement à l'avance
Une infrastructure fiable qui coûte très chère	La fiabilité est intégrée dans l'architecture du Cloud
Un environnement complexe	L'environnement est basé sur une architecture modulaire
Infrastructure complexe	Pas d'infrastructure

Plusieurs avantages ont rendu le Cloud Computing comme étant une solution très puissante à utiliser, surtout pour les start-ups. Parmi ces avantages, le coût d'investissement initial et de maintenance est très faible vu que tous les équipements et logiciels sont hébergés chez le fournisseur, donc tout coût relié à l'équipement est évité. De plus, les utilisateurs peuvent échelonner la taille des unités selon leurs besoins, ce qui rend la tâche du test lors du développement très rapide vu que les développeurs peuvent varier les unités des calculs selon leur besoin pour simuler les systèmes ciblés.

Chapitre II : Attaques et vulnérabilités

Grâce à la puissance du Cloud, plusieurs organisations ont préféré la migration vers cette technologie, mais 87.5% des organisations ont toujours peur de l'utiliser à cause de certaines raisons de sécurité.

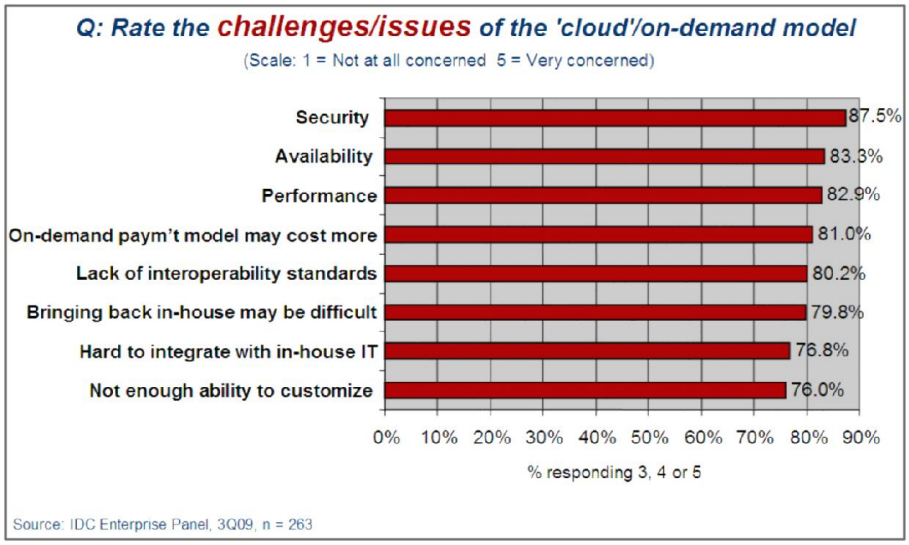


Figure II-1 : évaluation des défis du Cloud Computing par les entreprise

Dans ce chapitre, nous allons illustrer les différents problèmes dans les modèles de services, puis nous allons détailler les problèmes concernant la virtualisation et la sécurité des données, vu qu'elles représentent la partie vitale dans la sécurité du Cloud.

1. Vulnérabilité des modèles de services

Le modèle de service se compose de trois couches qui représentent le niveau de contrôle du client sur les données et les services exploités, ce qui signifie que chaque niveau hérite les problèmes et la vulnérabilité des niveaux inférieurs. Dans cette partie, nous allons présenter quelques vulnérabilités dans chaque niveau [10, 12-14].

1.1. SaaS (Software comme service)

La sécurité des applications est un élément critique de la sécurisation de n'importe quel système, mais différemment aux systèmes traditionnels, l'implémentation du Cloud nécessite une réévaluation des standards actuels dans le but de mettre en œuvre un système de sécurité qui gère les applications multi-utilisateurs.

Les applications fournies par le Cloud sont généralement des applications web exploitées à partir des navigateurs, ce type d'application à un grand nombre de vulnérabilité et de risque d'attaque de manières différentes. D'après un document publié par OWASP [15] en 2013 (Open Web Application Security Project) basé sur huit bases de données venant de sept entreprises, les dix risques les plus critiques sont :

- **Injection**

Une faille d'injection, telle l'injection SQL, OS et LDAP, se produit quand une donnée non fiable est envoyée à un interpréteur en tant qu'élément d'une commande ou d'une requête. Les données hostiles de l'attaquant peuvent duper l'interpréteur afin de l'amener à exécuter des commandes fortuites ou accéder à des données non autorisées.

- **Violation de gestion d'authentification et de session**

Les fonctions applicatives relatives à l'authentification et la gestion de session ne sont souvent pas mises en œuvre correctement, permettant aux attaquants de compromettre les mots de passe, clés, jetons de session, ou d'exploiter d'autres failles d'implémentation pour s'approprier les identités d'autres utilisateurs.

- **Cross-site Scripting (XSS)**

Les failles XSS se produisent chaque fois qu'une application accepte des données non fiables et les envoie à un browser web sans validation appropriée. XSS permet à des attaquants d'exécuter du script dans le navigateur de la victime afin de détourner des sessions utilisateur, défigurer des sites web, ou rediriger l'utilisateur vers des sites malveillants.

- **Références directes non sécurisées à un objet**

Une référence directe à un objet se produit quand un développeur expose une référence à un objet d'exécution interne, tel un fichier, un dossier, un enregistrement de base de données ou une clé de base de données. Sans un contrôle d'accès ou autre protection, les attaquants peuvent manipuler ces références pour accéder à des données non autorisées.

- **Mauvaise configuration de sécurité**

Une bonne sécurité nécessite de disposer d'une configuration sécurisée définie et déployée pour l'application, contextes, serveur d'application, serveur web, serveur de base de données et la plate-forme. Tous ces paramètres doivent être définis, mis en œuvre et maintenus, car beaucoup ne sont pas livrés et sécurisés par défaut. Cela implique de tenir tous les logiciels à jour.

- **Exposition de données sensibles**

Beaucoup d'applications web ne protègent pas correctement les données sensibles telles que les cartes de crédit, identifiants d'impôt et informations d'authentification. Les pirates peuvent voler ou modifier ces données faiblement protégées pour effectuer un vol d'identité, de la fraude à la carte de crédit ou autres crimes. Les données sensibles méritent une protection supplémentaire tel un chiffrement statique ou en transit, ainsi que des précautions particulières lors de l'échange avec le navigateur.

- **Manque de contrôle d'accès au niveau fonctionnel**

Pratiquement toutes les applications web vérifient les droits d'accès au niveau fonctionnel avant de rendre cette fonctionnalité visible dans l'interface utilisateur. Cependant, les applications doivent effectuer les mêmes vérifications de contrôle d'accès sur le serveur lors de l'accès à chaque fonction. Si les demandes ne sont pas vérifiées, les attaquants seront en mesure de forger des demandes afin d'accéder à une fonctionnalité non autorisée.

- **Falsification de requêtes intersites (CSRF)**

Une attaque CSRF (Cross Site Request Forgery) force le navigateur d'une victime authentifiée à envoyer une requête HTTP forgée, comprenant le cookie de session de la victime ainsi que toute autre information automatiquement inclue, à une application web vulnérable. Ceci permet à l'attaquant de forcer le navigateur de la victime à générer des requêtes dont l'application vulnérable pense qu'elles émanent légitimement de la victime.

- **Utilisation de composants avec des vulnérabilités connues**

Les composants vulnérables, tels que bibliothèques, contextes et autres modules logiciels fonctionnent presque toujours avec des privilèges maximums. Ainsi, si exploités, ils peuvent causer des pertes de données sérieuses ou une prise de contrôle du serveur. Les applications utilisant ces composants vulnérables peuvent compromettre leurs défenses et permettre une série d'attaques et d'impacts potentiels.

- **Redirection et renvois non validés**

Les applications web réorientent et redirigent fréquemment les utilisateurs vers d'autres pages et sites internet, et utilisent des données non fiables pour déterminer les pages de destination. Sans validation appropriée, les attaquants peuvent réorienter les victimes vers des sites de phishing ou de malware, ou utiliser les renvois pour accéder à des pages non autorisées.

Les hackeurs peuvent exploiter ces viabilités et plusieurs d'autres pour divers activités illégales comme la fraude financière, le vol de la propriété intellectuelle et les tentatives d'arnaques, surtout pour les applications déployées dans des Cloud publics qui sont exposées à un niveau élevé de risques.

1.2. PaaS (Plateforme comme service)

Le PaaS offre au développeur la possibilité de suivre le cycle de développement de leur application et de la déployer sur le serveur sous format d'un services SaaS, cela implique que la sécurité des applications SaaS dépend de l'expertise des développeurs au niveau PaaS ; des vulnérabilités ou des failles non prises en considération dans le développement d'une application peuvent mettre en risque les données utilisées dans le service ou même permettre aux développeurs malicieux d'injecter des scripts pour être exécutés.

1.3. IaaS (Infrastructure comme service)

Différemment au SaaS et au PaaS, la responsabilité du client augmente dans le cas de l'IaaS vu qu'il est responsable de tout ce qui se trouve au-dessus de la couche de la virtualisation, le client donne la responsabilité de tout ce que se trouve au-dessous du système d'exploitation au serveur (Réseaux, Virtualisation, Stockage) .

Quand on parle du niveau réseau, on doit faire la différence entre le Cloud public et le Cloud privé, car pour ce dernier, il n'y a pas de risque additionnel par rapport à la sécurité dans le modèle traditionnel. En effet lors de l'implémentation du Cloud privé il n'y a pas de changement majeur par rapport à l'architecture traditionnelle, cette l'architecture de Cloud est similaire à celle du extranet implémenté par les organisations aujourd'hui, ce qui est différent pour le Cloud public.

L'implémentation du Cloud public pour une organisation nécessite un grand changement dans la topologie du réseau, de plus l'organisation doit prévoir la méthode dans laquelle la communication entre le réseau interne de l'organisation et le fournisseur du Cloud public est établi. Dans ce type de scénario, il y a quatre grands facteurs de risque[10] :

1.3.1.1. Assurer une méthode d'accès appropriée

Lors de l'utilisation du Cloud public, l'organisation subit un risque majeur dans la sécurité, de plus la possibilité de faire l'audit de ressources fournies est presque inexistante, ainsi que l'accès aux fichiers log et la capacité de mener des enquêtes et de recueillir des données est très limitée.

Comme un exemple du problème d'accès, nous proposons la notion de réutilisation des adresses IP. Lors de la libération de l'adresse par l'utilisateur, elle est automatiquement associée à un autre utilisateur, mais cette adresse est toujours associée à l'utilisateur sans changer les données dans le DNS, ce qui fait que l'utilisateur courant peut avoir l'accès à des ressources qui sont supposées inaccessibles, cela est dû au fait qu'un temps est nécessaire avant la réinitialisation du DNS et la suppression du cache.

1.3.1.2. Assurer la disponibilité des ressources

Avec la notion des partages des ressources pour un Cloud public, les ressources sont partagées par tous les utilisateurs de ce dernier. De plus l'utilisation du Cloud est associée avec des risques de sécurité de ce niveau comme les attaques DDOS, IP Hijaking et les attaques de DNS.

1.3.1.3. Remplacer les niveaux et les zones de réseau établis avec des domaines

Dans le modèle traditionnel qui se base sur le concept des zones et de la séparation du réseau pour améliorer la sécurité, c'est-à-dire qu'une zone spécifique peut être accédée par des individus et des systèmes avec des rôles spécifiques, cette séparation n'était pas seulement logique, en effet des groupes de systèmes qui ont été séparé dans le modèle traditionnel logiquement et physiquement au niveau du hôte, cette séparation a été remplacée dans le Cloud

par des groupes de sécurité , domaines de sécurité et les centres de données virtuelles qui ont une séparation purement logique moins précise entre les niveaux et permettent moins de sécurité par rapport au modèle traditionnel, ce qui fait qu'un système de développement et un système de production peuvent être implémentés dans le même serveur, ce qui fait que dans le Cloud la notion de séparation existe seulement au niveau logique pour des raisons d'adressages, ce qui fait que deux systèmes peuvent être hébergés dans le même serveur avec une séparation virtuelle seulement.

Les problèmes de sécurité au niveau réseau sont les mêmes, peut import le type d'aspect de Cloud utilisé (IaaS, PaaS, SaaS), ce qui fait que la détermination de risque n'est pas liée au type de *aaS utilisé mais au type de déploiement implémenté (Public, Hybride, Privé).

1.3.1.4. La sécurité des logiciels de virtualisation

Les fournisseurs de Cloud gèrent la couche de virtualisation qui est directement au-dessous du hardware d'une façon que les utilisateurs n'ont aucun accès à cette couche, qui est essentielle vu qu'elle permet le partage des ressources entre multiples clients du Cloud sans interférence entre eux, permettant l'exécution de plusieurs systèmes d'exploitation dans le même ordinateur en même temps.

Vu son importance dans la compartimentation et dans l'isolation des clients des VMs entre eux dans un environnement multi-tenant, il est essentiel de protéger l'Hyperviseur de tous les utilisateurs non autorisés (voir **Sécurité de la virtualisation**), ce qui fait que les fournisseurs de services doivent prévoir les différents contrôles de sécurité et restreindre l'accès à cette couche pour garantir l'intégrité et la disponibilité des systèmes.

1.3.2. La gestion des serveurs virtuels

Les fournisseurs des IaaS publics offrent une API de web services pour effectuer les tâches de gestion pour assurer l'élasticité des ressources, ce qui leur permet d'élargir ou de rétrécir les ressources selon la demande, mais cette dynamité peut entrainer la complexité dans les cas de mauvaise configuration, de point de vu attaque, le serveur est accessible depuis internet par n'importe quel client ce qui fait qu'une restriction d'accès aux instances virtuelles est indispensable.

Dans la plupart des cas, les fournisseurs des serveurs virtuels bloquent tous les ports d'accès et recommandent l'utilisation du Shell sécurisé ou SSH (port 22) pour administrer le serveur virtuel, en effet il existe plusieurs risques de sécurité concernant cette partie comme :

- Le vol de la clé utilisée pour la gestion de l'hôte.
- Les attaques ciblant les vulnérabilités des services qui écoutent sur les ports standards comme FTP, NetBIOS, SSH….
- Le détournement des comptes qui ne sont pas bien sécurisés à cause de l'utilisation des mots de passes faibles.
- Attaques sur les systèmes mal sécurisés par l'hôte.

- Le déploiement des chevaux de Troie intégré dans des composantes de la machine virtuelle ou dans le système lui-même.

1.3.3.Attaques sur le réseau :

Le risque de sécurité au niveau du réseau est classé sous trois types ; garantir la confidentialité, la disponibilité et l'intégrité des données et les ressources auparavant confinées à un réseau privé, ces dernières sont désormais exposées à Internet.

En plus, le client/l'utilisateur qui utilise le protocole HTTP plutôt que HTTPS augmente les risques de sécurité. Par exemple Les attaques comme DOS[4] et DDOS[5] visent à perturber les services fournis par le Cloud. Ce type d'attaque est très difficile à détecter car les requêtes d'attaque sont mélangées avec le trafic, ce qui fait qu'une tentative de filtrer les ressources perturbera la qualité des services fournis. Ce type d'attaque est traditionnel, mais son impact sur le Cloud est différent. En effet ce type d'attaque impacte non seulement le trafic mais aussi la facturation de l'utilisation des ressources avec le principe de payer ce qu'on consomme du Cloud, cette attaque augmente la consomption des ressources (la bande passante, CPU…) on parle donc d'une d'attaque de type EDos.

2. Sécurité de la virtualisation

L'utilisation du virtualisation offre des possibilités techniques et économiques nouvelles mais, avec la puissance de cet outil, vient de nouveaux défis et vulnérabilités, voici une liste des défis soulevés par l'utilisation de la virtualisation [6, 8, 16] :

- Ecaillage : la virtualisation offre la possibilité de créer des nouvelles machines virtuelles, ce qui fait que les politiques de sécurité doivent avoir accès flexible pour pouvoir gérer la croissance dans le nombre de machines.

- Fugacité : le fait de pouvoir d'ajouter et de supprimer des nouvelles machines virtuelles rend la stabilisation plus difficile, cela est dû à la difficulté de la détection des machines infectées, car il n'existe qu'à pour une brève période.

- Le cycle de vie logiciel : le fait de pouvoir rétablir une machine virtuelle à un état antérieur donne la possibilité d'exploiter des failles déjà patchées ce qui augmente le risque de sécurité et rend les protocoles de sécurité établis pour une machine avant la restauration obsolète.

[4]DOS : Une « attaque par déni de service » est un type d'attaque visant à rendre indisponible, pendant un temps indéterminé.

[5]DDOS : est une attaque de déni de service prévenant de plusieurs sources en même temps, ce type d'attaque est plus puissant que DOS.

- La mobilité et l'identité : une machine virtuelle n'est qu'un fichier ou un dossier dans le disque dure du serveur ce qui fait qu'il est possible de déplacer ce VM à un autre host, ce qui rend la sécurisation de ce système relier à la sécurisation du host ou des hosts dans le(s) quel(s) il est hébergé, de plus cette mobilité rend la vérification de l'authenticité des systèmes très difficile surtout que les méthodes traditionnelles d'identification des machines ne peuvent pas nécessairement être efficaces dans le cas des machines virtuelles.

- Le cycle de vie de donnée : le fait qu'un hyperviseur peut garder l'état des machines virtuelles peut rendre les efforts de suppression des données sensibles futiles, car une version de sauvegarde peut exister.

Techniquement parlant, une machine virtuelle est considérée comme des machines physiques ce qui fait qu'il est possible d'utiliser les mêmes méthodes pour contrôler les machines, en plus l'utilisation du concept de la virtualisation donne naissance à des nouvelles méthodes pour la subversion des machines virtuelles.

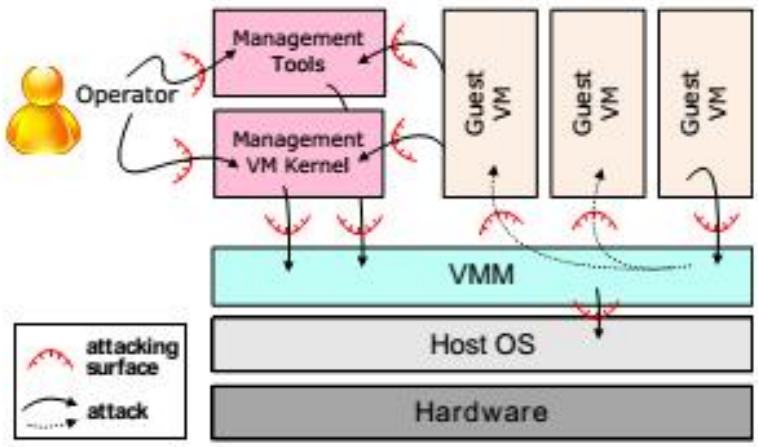


Figure II-2: Attaques sur les machines virtuelles

Comme illustré dans la figure II-2, les failles dans la couche de virtualisation permettent de lancer une attaque de la machine virtuelle vers une autre, vers l'hyperviseur ou vers le système d'exploitation qui se trouve au-dessous, le schéma dans cette figure correspond au cas d'un hyperviseur de type 2, ce qui est similaire au type 1, la seule différence est l'inexistence du système OS au-dessous de l'hyperviseur (VMM), ce qui fait que la surface d'attaque correspondante sera supprimée aussi, on peut donc classifier les attaques sur la virtualisation sous trois catégories [6, 8] :

- Depuis la machine virtuelle vers l'hyperviseur,
- Depuis l'hyperviseur vers la machine virtuelle,
- D'une machine virtuelle vers une autre.

2.1. Depuis la machine virtuelle vers l'hyperviseur

Ce type d'attaques est le plus dangereux et représente le pire cas dans le scénario pour la technologie de virtualisation, car il permet à l'attaquant de lire/écrire/modifier les données et les programmes, même s'il n'as pas le droit de le faire, l'effet de cette attaque n'est pas limité à la machine virtuelle mais il impacte les autres machines et l'hyperviseur lui-même.

Ce type d'attaques donne plusieurs possibilités à l'attaquant, il peut par exemple modifier les privilèges de sa machine virtuelle pour la rendre plus privilégiée il peut aussi espionner sur les autres machines tout en restant invisible.

2.1.1. Direct accès à la mémoire

Il existe plusieurs drivers qui ont la possibilité d'accéder au matériel (réel), cet accès est réalisé à travers DMA (direct Memory Access), ce qui donne la possibilité de lire et écrire dans la mémoire physique à l'attaquant qui exploite ses bugs.

Normalement dans une architecture virtualisée l'accès à la mémoire est contrôlé par l'hyperviseur qui garantit que chaque machine virtuelle a l'accès à sa propre plage mémoire, mais un attaquant peut accéder à la totalité de la mémoire physique (comme illustré dans la figure II-3) en exploitant les bugs du DMA qui relient l'interface de réseau virtuel avec la carte réseau réel, cette dernière a un accès direct à la mémoire sans aucun contrôle de la part de l'hyperviseur.

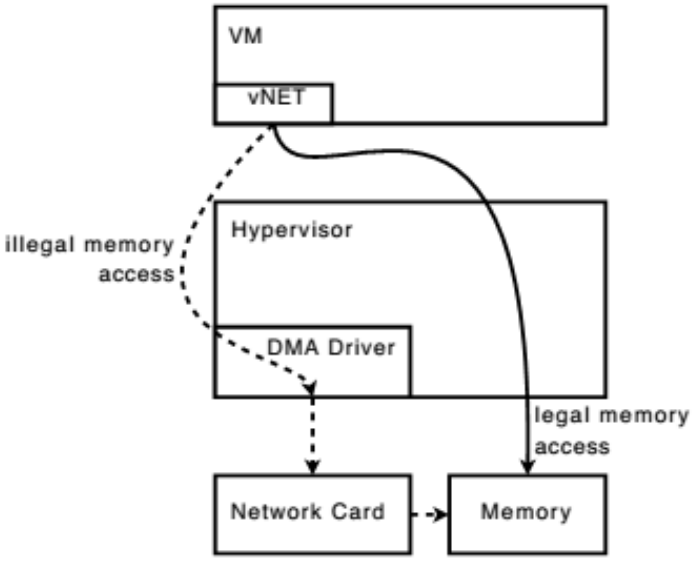


Figure II-3: accès malicieux au mémoire à travers le DMA[6]

Ce type d'attaque peut corrompre l'intégrité et la confidentialité de l'hyperviseur et de toutes les machines virtuelles qui s'exécutent dans cette machine, en plus l'attaquant peut provoquer un statut de DOS en supprimant le noyau d'une machine virtuelle à partir de son espace mémoire.

2.1.2. Autre vecteurs d'attaques

Certains matériels dédiés à la virtualisation contiennent des instructions qui ont créé d'autre vecteurs d'attaques, par exemple les processeurs de Intel contiennent une instruction qui permet la reconfiguration de la mémoire, un attaquant peut exploiter cette instruction pour migrer une partie de sa zone mémoire vers une zone dans laquelle il peut lire, écrire et modifier ce qui crée un attaque similaire à celui présenté précédemment sans passer par DMA.

En plus, la complexité croissante des hyperviseurs rendent la validation de code de plus en plus difficile, ce qui peut donner des nouveaux vecteurs d'attaques, par exemple si on peut exploiter un « buffer overflow » d'un hyperviseur on peut exécuter un code avec les privilèges de l'hyperviseur.

2.2. De l' hyperviseur vers la machine virtuelle

Les attaques de ce type ont comme but de perturber, espionner, voler ou modifier les machines virtuelles avec un code malicieux exécuté au niveau de l'hyperviseur, ce type d'attaques est basé sur l'injection du code dans hyperviseur, cette injection est faite soit avec la méthode décrite précédemment ou par un programme malicieux exécuté dans le même niveau que l'hyperviseur, « Blue Pillattack » est un exemple très connu de cette catégorie.

Cette catégorie peut être divisée en trois sous-catégories [6] :

- Hardware accelerated rootkits (HAR)
- Machine virtuelle à grand privilège
- Attaques sur le hardware.

2.2.1. Hardware accelerated rootkits (HAR)

Ce type d'attaque peut virtualiser un système d'exploitation ou un hyperviseur en cours d'exécution sans être détecté, ce qui donne la possibilité d'avoir un hyperviseur non visible de la part de l'utilisateur qui a la possibilité d'espionner et de contrôler la machine virtuelle toute entière, ce type d'hyperviseur est connu sous le nom « HAR ».

Ce type d'attaques est possible de plusieurs manières :

1- En construisant un hyperviseur et le positionner dans une couche au-dessous d'un autre hyperviseur avec la manière indiquée dans la figure II-4.

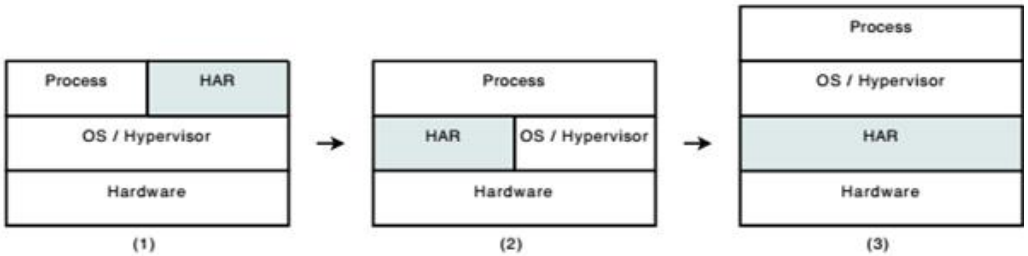


Figure II-4: HAR qui prend le contrôle du système[6]

2- En modifiant l'hyperviseur d'une manière statique pour qu'il fasse des opérations malicieuses, cette forme d'attaque est difficile à détecter.

3- En modifiant l'hyperviseur d'une manière dynamique, cela est possible avec DMA ou autre vecteurs d'attaque permettant la modification de la mémoire pour remplacer le modèle de hyperviseur avec un autre malicieux. Parmi les trois formes de cette attaque celle-ci est la plus difficile à détecter, car de point de vue du système rien n'est changé au cours et après de la virtualisation.

2.2.2. Machine virtuelle à grand privilège

Dans le cadre de protection des machines virtuelles, des hyperviseurs qui offrent la possibilité d'exécuter des machines virtuelles dans des niveaux de privilèges différents ont été développé pour permettre de créer une machine virtuelle à grand privilège, permettant d'effectuer des scans sur l'ensemble de machines virtuelles pour détecter les anomalies (le cas d'un antivirus, IDS, IPS) et qui ont aussi une interaction directe avec le matériel. Xen donne la possibilité de créer une machine virtuelle avec l'un des deux niveaux de privilèges :

- DomU : ce sont les machines virtuelles qui ont un niveau de privilège normal, en conséquent quelque-soit l'interaction de ce type de machine avec le matériel, cette machine doit obligatoirement passer pas une machine virtuelle à haute privilège.

- Dom0 : ce sont les machines virtuelles à haute privilège, qui ont accès au matériel, ce qui fait ces machines ont la possibilité de lire et écrire dans la mémoire de la totalité des machines virtuelles.

Comme illustré dans la figure I-5, dans ce type d'architecture il existe trois niveaux, niveau utilisateur (ring 3) qui ont besoin d'interagir avec l'hyperviseur à chaque fois qu'il a besoin d'avoir accès à la mémoire, le niveau des machines privilège (ring 0) qui ont, comme le niveau hyperviseur (ring -1), un accès direct à la mémoire.

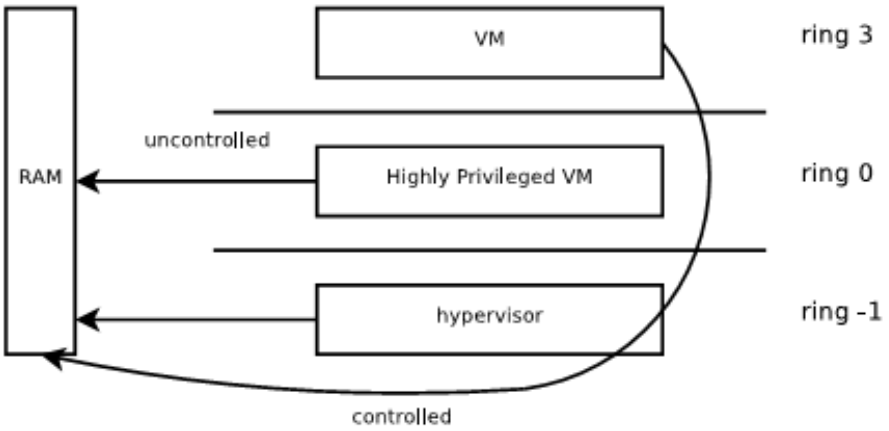


Figure II-5: l'accès au mémoire depuis une machine privilégié et une autre normal[6]

Cette architecture a donné naissance à un nouveau vecteur d'attaque qui distingue dans la création une machine virtuelle à grand privilège avec des intentions malicieuses, avec son accès libre à la mémoire, cette machine peut espionner, lire, écrire et modifier les données des autres machines virtuelles, ce qui met la confidentialité, l'intégrité et la disponibilité des autres machines virtuelles, en plus cette attaque ne peut pas être détectée au niveau d'autres machines virtuelles, ce qui rend la détection possible seulement par un scan du part de l'hyperviseur pour des virtuelles machines à hautes privilèges qui espionnent sur les autres virtuelles machines, mais cette possibilité peut être dépassée en enregistrant ces virtuelles machines comme des composantes de sécurité, ce qui rend les scans (espionnage) sur les autres machines virtuelles une opération normale.

2.2.3.Attaques sur le hardware

L'idée principale de la virtualisation c'est de permettre à plusieurs systèmes d'exploitation d'être exécutés sur le même matériel, mais cette possibilité elle-même donne une autre possibilité de créer des flux d'informations entre l'hyperviseur et les machines virtuelles, ce qui fait que tous les matériels partagés sont des sources potentielles des attaques provenant de l'hyperviseur.

- VHDD (les disques durs virtuels)

  Normalement un disque dur virtuel peut être soit un fichier dans un disque dur, une partition d'un disque physique ou un espace de stockage réseau. Pour ces trois types de VHDD un hyperviseur peut facilement y accéder à cause de son niveau de privilège, ce qui met l'intégrité et la confidentialité des données en risque.

- VRAM (mémoire virtuelle)

  Un hyperviseur à un accès direct à la mémoire ce qui fait que la confidentialité, l'intégrité et la disponibilité des données dans une VRAM peuvent être facilement attaquées par un hyperviseur malicieux ou une virtuelle machine à haut privilège malicieuse.

- VCPU (processeur virtuel)

  Les données d'un VCPU peuvent être facilement lues par des instructions matérielles prévenant d'une source privilège comme un hyperviseur, en plus un attaquant peut accéder aux registres des VCPU dans le processus de vérification de l'intégrité et retourner une réponse positive peu importe le contenu du fichier vérifié, ce qui permet potentiellement de lancer n'importe quel code dans la machine virtuelle.

- VNET (réseau virtuel)

  Dans la majorité des implémentations des hyperviseurs, les cartes réseau fournies au machines virtuelles ne sont que des ponts logiciels qui sont connectés à la carte réseau physique cela donne la possibilité a un hyperviseur malicieux de sniffer et d'attaquer la confidentialité des données des machines virtuelles, en plus ce dernier peut

modifier ces ponts logiciels pour le contenu des paquets transféré dans le réseau virtuel ce qui va mettre en risque l'intégrité du VNET de n'importe quelle machine virtuelle.

Tous les attaques présentées précédemment sont dues au fait que l'hyperviseur a un niveau de privilèges supérieur à celui des machines virtuelles sur le niveau matériel, ce qui rend ces attaques toujours possibles tant que l'hyperviseur n'est pas contrôlé par une autre partie ou entité approuvée.

2.3. D'une machine virtuelle vers une autre

Le but de ce type d'attaque c'est de perturber une machine virtuelle à partir d'une autre, il existe deux types d'attaques dans la littérature, attaques sur les ressources et le détournement (backdoors).

2.3.1. Attaques sur les ressources

Presque toutes les ressources partagées peuvent être utilisées pour ce type d'attaques, le but principal c'est de modifier les ressources pour attaquer les machines virtuelles, l'utilisation des DMA est un exemple de cette modification. Comme expliqué précédemment, l'attaquant peut accéder aux mémoires des autres virtuelles machines en passant par DMA ensuite lire et écrire au-dessus.

2.3.2. Attaques à détournement (Backdoors) :

Un Backdoor est un spécial piège inséré dans un programme exécuté dans une machine pour permettre l'accès au système sans être détectable, par défaut ces pièges sont désactivés et réagissent à des événements bien précis pour ouvrir une « porte » caché dans le système, l'utilisation de ce type d'attaques nécessite la modification des hyperviseurs pour qu'il réagisse à un évènement en entrant dans un mode spécial permettant l'espionnage d'une machine sur une autre ou en augmentant les privilèges du machine virtuelle de l'attaquant.

Ce type d'attaques met la confidentialité, la disponibilité et l'intégrité des machines virtuelles en risque, car une attaque de ce type peut facilement modifier les données d'authentification dans les fichiers de configuration des virtuelles machines, ce qui permet l'authentification à n'importe quelle machine virtuelle en mode root.

3. Sécurité des données

Le principe de sécurité de données dans le Cloud est similaire à celui dans les architectures traditionnelles, cette confidentialité est impliquée dans toutes les étapes du cycle de vie de l'information, mais grâce à la nature multi-tenante du Cloud cette sécurité acquiert certaines particularités.

La sécurité des données doit être assurée dans toutes les étapes du cycle de vie de l'information (figure I-6) du génération jusqu'à la destruction [17, 18] :

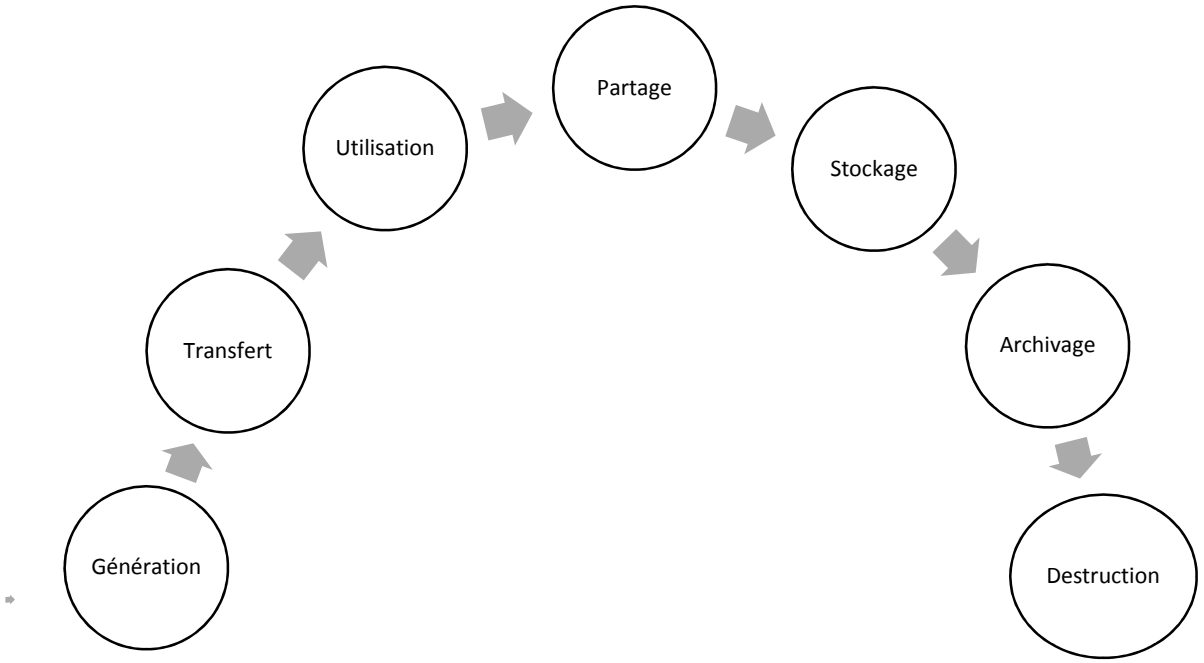


Figure II-6: Cycle de vie de l'information

3.1. Génération

Lors de la migration des données chez un fournisseur du Cloud, le client doit avoir les informations et garantir que la propriété des données sera conservée lors de toutes les étapes du cycle de vie d'information et la possibilité de contrôler les informations collectées par le fournisseur de service.

3.2. Transfert

Lors du transfert des informations, les données sont transférées à travers l'internet ce qui fait qu'il y a des risques de lecture et de modification des données au cours de ce processus, ce qui fait que le fournisseur d'accès doit fournir une méthode de transfert qui garantit la confidentialité et l'intégrité des informations, cette méthode ne doit pas seulement être appliquée entre le fournisseur et le client mais aussi lors de transfert inter-Cloud des informations.

3.3. Utilisation

Pour les services simples comme dans le cas de stockage d'information, il est facile de traiter le problème de confidentialité des données par une simple implémentation des algorithmes de cryptages, ce qui est infaisable dans le cas où les services sont orientés traitement car le cryptage dans ce type de service crée un problème d'indexation et d'interrogation, ce qui fait que les données reliées à ce type de service sont généralement stockées sans cryptage avec les données des autres clients, ce qui pose des risques de sécurité.

3.4. Partage

La possibilité de cryptage total ou partiel d'information augmente la complexité du processus de gestion des autorisations.

3.5. Stockage

Similaire aux architectures traditionnelles le stockage de données dans le Cloud doit assurer les aspects de confidentialité, intégrité et la disponibilité.

3.5.1. Confidentialité

Comme indiqué précédemment dans le cas des services orientés stockage, la confidentialité peut être assurée grâce aux algorithmes de cryptographie, mais la nature partagée des ressources dans le Cloud impose des limitations de ressource et de temps au type des algorithmes implémentés, en plus d'une gestion optimale des clés.

Idéalement les clés doivent être gérées par les clients mais à cause du manque d'expertise le client confie cette tache aux fournisseurs du service ce qui nécessite un degré de confiance entre le client et le fournisseur.

3.5.2. Intégrité

Un autre problème dans le Cloud est relié à l'intégrité, pour un utilisateur qui a plusieurs GB d'information dans le Cloud, la vérification de l'intégrité est difficile, l'utilisateur doit avoir la possibilité de vérifier l'intégrité sans être obligé de télécharger toute l'information la vérifier puis la ré-uploader chez le fournisseur, ce qui consomme le temps, la bande passante et (chez quelques fournisseurs comme Amazone) les frais.

3.5.3. Disponibilité

Dans les architectures traditionnelles la question de la disponibilité était reliée principalement aux attaques externes, ce qui n'est pas le cas dans le Cloud, en plus des attaques externes, la disponibilité dans cette technologie est liée à :

- La disponibilité des services fournis par le Cloud : les services de Cloud peuvent facilement tomber hors service à cause d'une attaque externe (comme DOS) ou interne (les attaques sur la virtualisation).
- Le fait que le fournisseur de Cloud continuera à fonctionner dans l'avenir ou pas : le client doit savoir si dans le cas où un fournisseur cesse de fonctionner, les informations stockées dans le Cloud seront-elles récupérables ou transférables vers un autre ou récupérables d'un moyen ou d'un autre.
- Le fait que ce fournisseur du service de stockage fournit le service de la sauvegarde (backup).

3.6. Archivage

Ce processus se page généralement sur les supports de stockage, ce qui fait que si un fournisseur de service Cloud utilise un support de stockage externe un risque de fuite de données est imposé et si le fournisseur n'utilise pas ces supports un risque lié à la disponibilité apparait.

3.7. Destruction

Pour des raisons reliées au support de stockages même après la suppression des données il y a une possibilité que les données existent toujours.

Chapitre III : Sécurité du Cloud Computing

1. Introduction

1.1. Objectif de la sécurité

La sécurité permet de garantir la confidentialité, l'intégrité, l'authenticité et la disponibilité des informations.

1.1.1. La confidentialité

La confidentialité assure que les données d'un client ne soient accessibles que par les entités autorisées, les fournisseurs du Cloud doivent fournir des outils pour permettre le stockage et l'utilisation de leurs données tout en conservant la confidentialité des données.

1.1.2. L'intégrité

Les clients qui cherchent à externaliser leurs données peuvent évidemment s'attendre à être protégés contre les modifications non autorisées.

1.1.3. La disponibilité

Les services fournis par les fournisseurs de Cloud sont accessibles via internet, ce qui signifie, pour un client qui externalise ses applications et ses ressources, les données et les services doivent être accessible à la demande. Pour cette raison de nombreux fournisseurs de Cloud comme Windows azure garantissent la disponibilité de ces ressources en se basant sur le principe de redondance [19], en dupliquant leur ressources sur différent nœud pour remédier aux pannes matérielles ce qui permet de garantir un niveau élevé de disponibilité.

1.2. Notion de base de la sécurité

1.2.1. Cryptographie symétrique et asymétrique

Deux familles de cryptographie existent depuis les années 1970. Elles se distinguent en fonction du type de clés utilisées. La cryptographie symétrique nécessite que les systèmes de chiffrement et de déchiffrement disposent de la même clé cryptographique, tandis que la cryptographie asymétrique ou à clés publiques considère deux clés complémentaires « les clés publique et privée » réalisant indifféremment l'une le chiffrement et l'autre le déchiffrement. Ces deux familles sont ci-après décrites avec quelques exemples d'algorithmes couramment utilisés de nos jours, leurs avantages et inconvénients ainsi que leurs complémentarités.

1.2.1.1. Cryptographie symétrique

La Cryptographie symétrique se base sur l'usage d'une même clé pour chiffrer et déchiffrer des données. Ces clés sont appelées des clés symétriques (parfois secrètes). Dans le cadre d'échanges sur un réseau, une entité émettrice chiffre les données avec une clé et l'entité destinatrice déchiffre les données avec la même clé. Partager une clé avec chaque entité communicante potentielle, même dans un groupe fermé d'entités est extrêmement

contraignant et conduit rapidement à un très grand nombre de clés à gérer. Il est donc préférable d'automatiser la mise en place de ces clés.

Les algorithmes symétriques les plus connus sont dans l'ordre chronologique de définition, le DES (Data Encryption Standard), le 3DES (prononcé « Triple DES ») et l'AES (Advanced Encryption Standard).

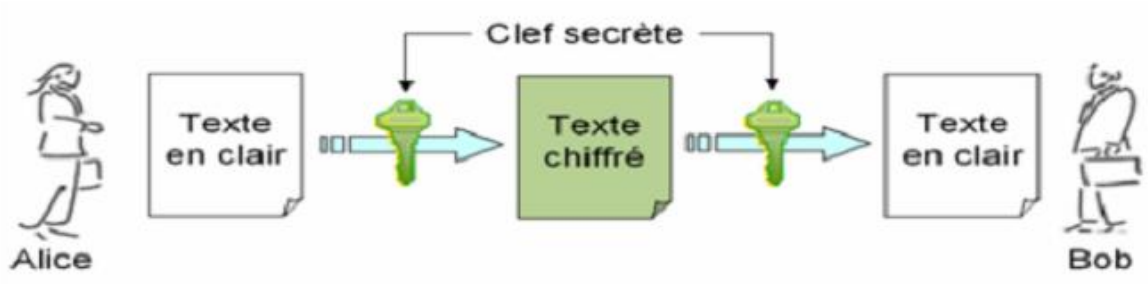


Figure III-1: Chiffrement Symétrique [1]

1.2.1.2. Cryptographie asymétrique ou à clés publiques

La cryptographie asymétrique ou à clé publique considère deux clés de chiffrement, dites « clés asymétriques ». Ces deux clés sont générées simultanément et sont complémentaires car le chiffrement avec l'une de ces clés nécessite le déchiffrement avec l'autre clé. Chaque clé a un rôle bien défini. La clé privée est une clé qui ne doit être connue que d'une seule entité, c'est elle qui permettra à cette entité de s'authentifier par exemple. Il est préférable que la clé publique soit largement diffusée, pour que l'entité concernée puisse être authentifiée par un grand nombre d'entités. Bien entendu, la connaissance de la clé publique ne doit pas permettre de déduire la clé privée complémentaire.

- **Chiffrement :**

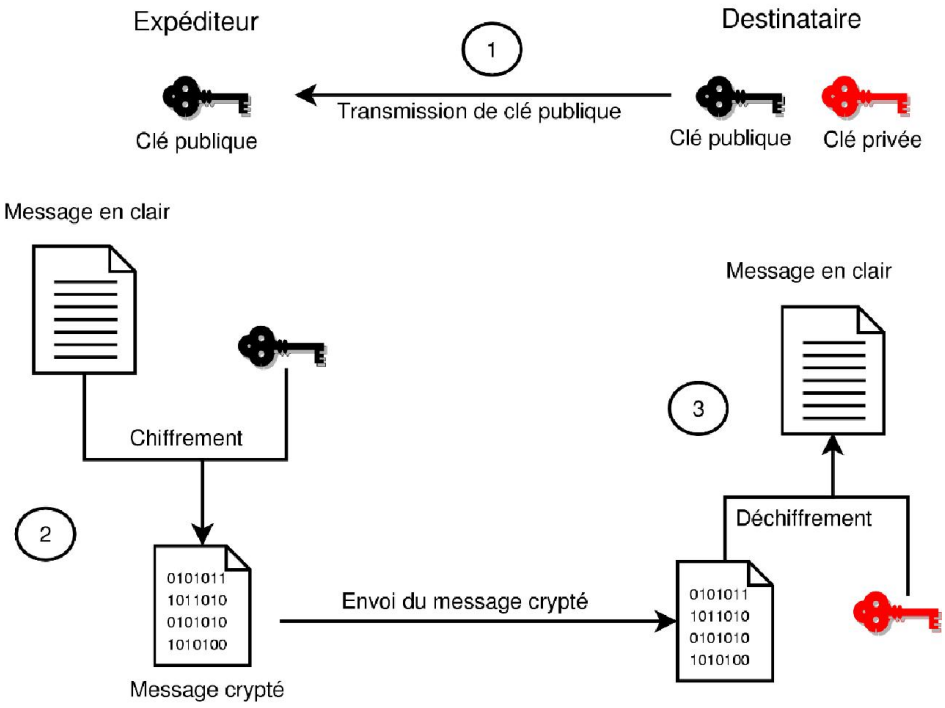


Figure III-2: Cryptographie asymétrique (Chiffrement) [1]

- **Signature**

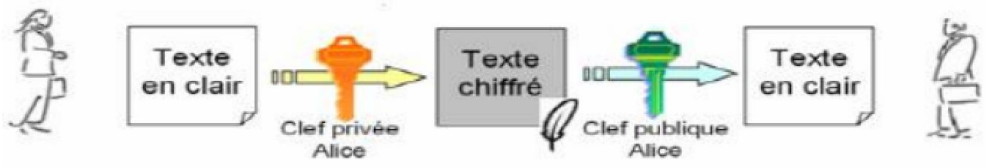


Figure III-3: Cryptographie asymétrique (Signature) [1]

1.2.2. Fonctions de hachage

Les propriétés attendues de ces fonctions de hachage sont les suivantes :

- Un résultat sur un nombre limité d'octets (en général 16 ou 20 octets).
- L'impossibilité de retrouver le message original à partir du résultat de la fonction.
- Deux messages différents de 1 bit seulement produisent deux résultats qui diffèrent d'au moins la moitié des bits.

Plusieurs termes désignent ces mêmes fonctions de hachage, à savoir : fonctions irréversibles, ou fonctions à sens unique. De même, plusieurs termes désignent le résultat de cette fonction appliquée à un message : hash, haché, empreinte, condensat ou encore condensé.

1.2.3. Signatures électroniques et MAC

La signature électronique ou le MAC (Message Authentication Code) apposée à un message a un double objectif: elle permet au destinataire d'authentifier l'origine de ce message et aussi de prouver son intégrité. Leur implémentation fait appel aux fonctions de hachage et aux clés symétriques ou asymétriques. Dans le cas de l' usage de la cryptographie symétrique, on emploie exclusivement le terme de MAC, tandis que dans l'usage de la cryptographie asymétrique, on peut parler de MAC, mais on préférera le terme de signature électronique.

Dans ce paragraphe, nous présentons les deux manières de générer un MAC, puis de vérifier la validité d'un MAC en fonction du type de cryptographie utilisée. Dans le cas de la cryptographie symétrique, comme le montre la figure III-4 et III-5, la génération du MAC suppose les opérations 1 à 4 de la part de l'émetteur (A) (figure III-4), tandis que la vérification par le récepteur (B) (figure III-5) de sa validité nécessite les étapes 5 à 9.

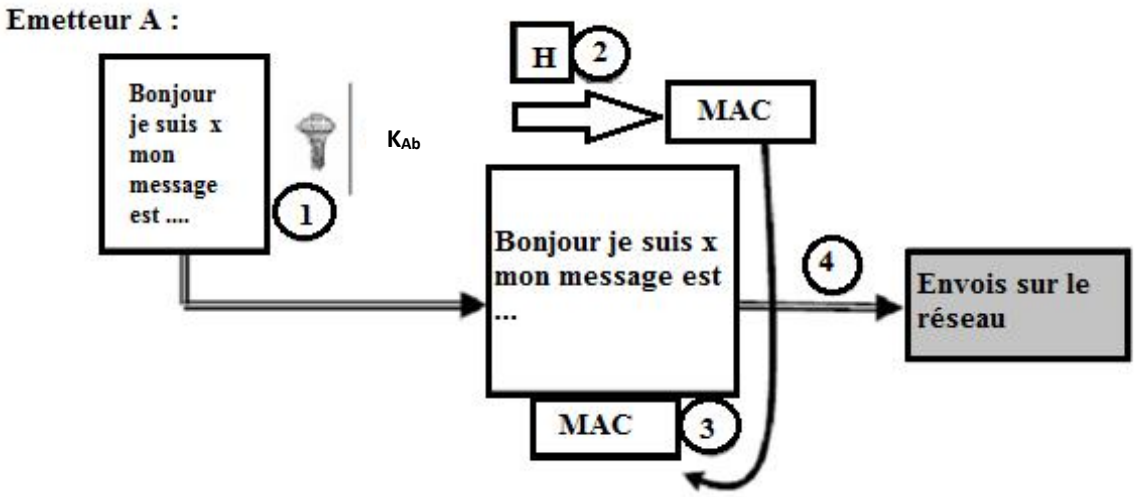


Figure III-4:Génération d'un MAC (Cryptographie symétrique)[1]

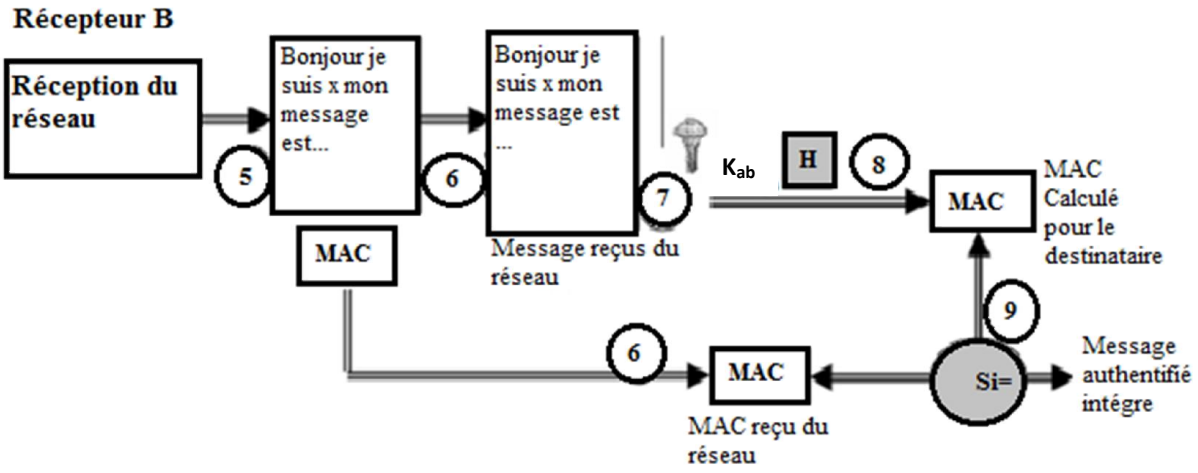


Figure III-5 : Vérification d'un MAC (Cryptographie Symétrique)[1]

L'émetteur tout d'abord combine le message et la clé KAB. Le résultat de cette première étape se trouve alors haché par la fonction de hachage H et le MAC est alors obtenu (étape 2) et apposé au message émis sur le réseau (étape 3) avant émission sur le réseau (étape 4). A la réception du message (étape 5), le récepteur sépare le message du MAC (étape 6). Sur le message reçu, un MAC est alors calculé localement en suivant les mêmes étapes que l'émetteur (étapes 7 et 8). Le MAC calculé en local est ensuite comparé au MAC reçu du réseau (donc logiquement celui calculé par l'émetteur s'il est intègre). En cas d'égalité (étape 9), le message reçu peut être considéré comme authentique et intègre. En effet, d'une part, le calcul du MAC faisant intervenir la clé partagée KAB, nécessairement l'émetteur d'un tel MAC ne peut être que l'émetteur déclaré, sinon l'entité ne connaîtrait pas la bonne clé. D'autre part, en cas de modification du message ou du MAC lors du transfert sur le réseau, il est clair que le MAC calculé par le récepteur serait différent du MAC reçu et il n'y aurait pas d'égalité.

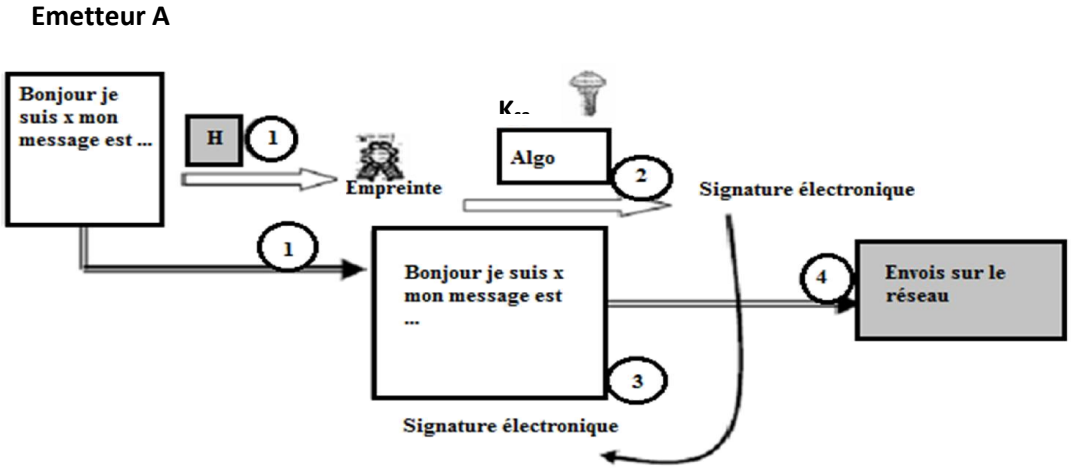


Figure III-6: Génération d'une signature électronique (Cryptographie asymétriques)[1]

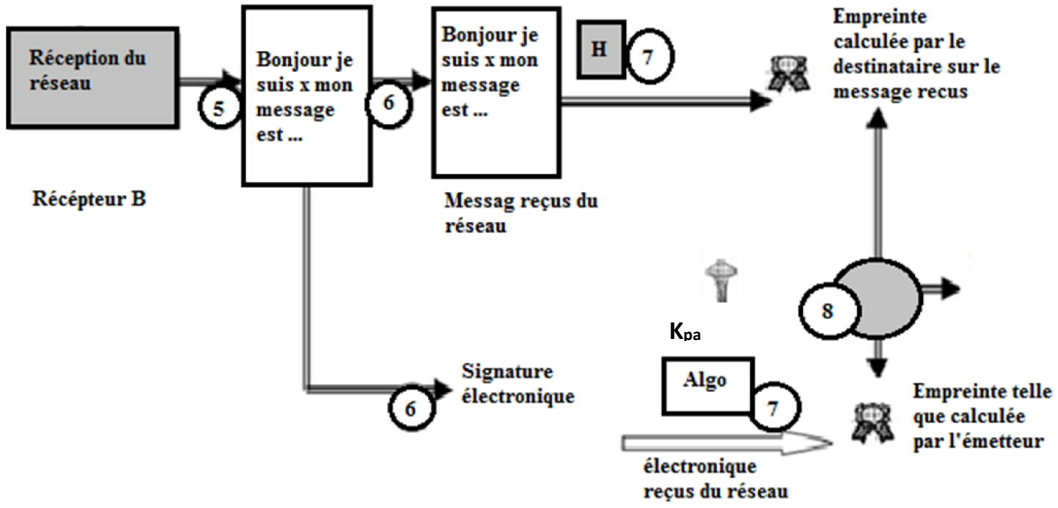


Figure III-7: Vérification d'une empreinte électronique (Cryptographie Asymétrique) [1]

Dans le cas de la cryptographie asymétrique, comme illustre la figure III-6 et III-7, l'émetteur a commencé par générer une empreinte à l'aide de la fonction de hachage H (étape 1), puis chiffre cette empreinte avec un algorithme asymétrique de sa clé privée (étape 2). Il obtient alors une signature électronique qu'il appose au message original (étape 3) avant d'émettre l'ensemble : message et signature sur le réseau (étape 4). A la réception (étape 5) (figure III-7), le récepteur B sépare le message de la signature (étape 6). Dans l'étape 7, d'une part, il calcule l'empreinte du message reçu localement et d'autre part, il déchiffre la signature reçue à l'aide du même algorithme de déchiffrement et de la clé publique de A pour obtenir l'empreinte telle que l'avait calculée l'émetteur. En cas d'égalité (étape 8), le message est prouvé authentique et intègre. Pour preuve, l'entité génératrice de la signature reçue doit nécessairement posséder la bonne clé privée KSA. Donc il ne peut s'agir que de A.

De plus, en cas de modification du message transmis sur le réseau, il est clair que l'empreinte calculée localement par le destinataire aurait abouti à un résultat très différent de celle calculée par l'émetteur (obtenue après déchiffrement de la signature par le récepteur).

1.2.4. PKI (Public Key Infrastructure)

Une infrastructure de gestion de clés (IGC) ou PKI (public key infrastructure) [20] prend en charge : la génération de clés publiques/privées et leur distribution à leurs propriétaires à l'initialisation d'une nouvelle entité dans la PKI, ainsi que la publication, révocation et validation de clés publiques. Généralement, les PKIs se basent sur des certificats électroniques et des listes de certificats révoqués, mais parfois, la simple publication de clés publiques de façon sécurisée dans un annuaire peut suffire. Parfois aussi, le jeu de clés est généré par son propriétaire qui ne requiert alors de la PKI que la certification de sa clé publique.

Les PKI distinguent les deux rôles d'autorités suivants :

- Autorité de certification (AC) : l'autorité de certification est la seule autorité à détenir la clé privée de l'autorité de certification et donc habilitée à émettre des certificats électroniques et des listes de certificats révoqués.
- Autorité d'enregistrement : Elle se charge de filtrer les demandes de certificat en effectuant un contrôle plus ou moins strict sur l'identité du demandeur, elle se charge aussi de publier et valider les certificats électroniques générés par l'autorité de certification.

1.2.5. Certificats électroniques

Les certificats électroniques ont pour objectif de lier de façon sûre une clé publique à une entité (utilisateur, serveur, etc.). Ces certificats correspondent concrètement à une structure de données dont le format le plus courant est fourni par le standard X.509v3 [20] et comprend entre autres : un numéro de série, une clé publique, l'identifiant du propriétaire de la clé publique, la date de validité (date de début et date de fin de validité), l'identifiant de l'autorité de certification (AC) émettrice du certificat, la signature du certificat à l'aide de la clé privée de l'autorité de certification. C'est la signature apposée par l'AC qui garantit l'authenticité du certificat. En effet, il suffit qu'une entité ait confiance dans l'AC et dispose de sa clé publique pour qu'elle puisse en confiance utiliser des clés publiques gérées par l'AC.

2. Sécurité de la virtualisation

L'Hyperviseur est un des éléments critiques dans la virtualisation avec son niveau de privilèges sa sécurisation est essentielle pour la sécurité des machines virtuelles.

2.1. Hyperviseur hébergé (type II)

Ce type d'hyperviseur est un processus dans l'os lui-même, ce qui fait que sa sécurisation peut être faite d'une manière normale ou traditionnelle en utilisant des antivirus, anti-malware [6].

Le problème de cette approche c'est qu'il est possible de n'importe quelle application exécutée sur l'os d'avoir le même niveau de privilège ou même plus que l'hyperviseur, ce qui permet de scanner et de modifier l'hyperviseur.

2.2. Hyperviseur natif (type I)

Ce type d'hyperviseur s'exécute directement au-dessus du niveau matériel ce qui rend le fait de réduire le nombre de lignes des codes très important pour faire la vérification et classifier l'hyperviseur comme étant fiable, ce qui rend le fait d'intégrer le processus de vérification dans hyperviseur lui-même très délicat, pour cette raison d'autres méthodes sont introduites [6]:

2.2.1. Composantes de sécurité classiques

Dans les approches classiques, on se basait sur l'antivirus et les anti-malware pour garantir la sécurité des systèmes, ce qui n'est pas faisable dans le cas des hyperviseurs pour deux raisons : la première c'est que l'hyperviseur n'est pas un système d'exploitation, ce qui rend certaines approches inapplicables, la deuxième c'est que le fait d'intégrer ce genre de processus dans les lignes de code de hyperviseur accroit la difficulté de vérifier son intégrité.

2.2.2. Réduire TCB avec la décomposition

Trusted computing base (TCB) représente l'ensemble des éléments systèmes et des processus essentiels dans la sécurité, les processus de ce type ont la possibilité de violer les règles de contrôle d'accès dans le système ce qui rend la sécurisation de ce type de processus une tâche critique dans la sécurisation de n'importe quel système.

Comme mentionné précédemment, pour permettre la vérification de l'hyperviseur, on doit avoir le minimum de lignes de codes possibles, le but de la décomposition c'est de réduire le nombre des TCBs dans l'hyperviseur, ces TCBs sont par suite mis dans des conteneurs avec moins de privilèges, comme illustré dans la figure III-8. Il existe trois manières pour réduire le nombre des TCBs à ce qui est essentiel :

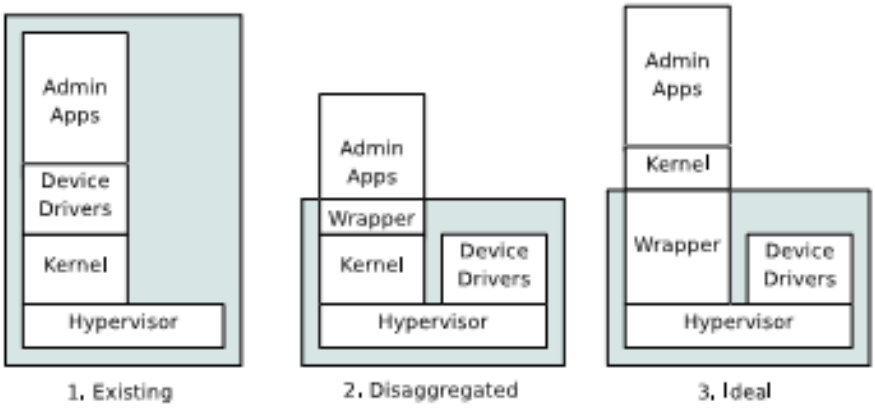


Figure III-8 : les trois model de virtualisation des décompositions TCBs[6]

1- Le premier modèle représente le modèle existant dans lequel le TCB englobe la totalité du système d'exploitation.
2- Le deuxième modèle représente un modèle dans lequel seulement le noyau et les pilotes nécessaire sont englobés dans le TCB ce qui fait de ce modèle une version meilleure que le premier.
3- Le modèle idéal est représenté dans le troisième modèle dans lequel seulement les pilotes sont considérés comme fiable, le conteneur (wrapper) représente le lien entre l'hyperviseur et ses outils de gestion ce qui permet la vérification des entrées et des sorties données aux outils de gestion.

Le problème dans cette approche c'est le nombre de communications important entre les domaines privilèges et les autres domaines, qui sont très coûteux en termes d'interruption des processus.

2.3. Vérification d'intégrité

Une autre façon pour garantir la sécurité des hyperviseurs c'est de vérifier leurs intégrités [6], cela est fait en comparant l'empreinte de l'hyperviseur au démarrage avec l'empreinte laissée lors de l'installation ce qui permet de détecter les modifications apportées au système.

Cette vérification peut être faite d'une moyenne logicielle en intégrant un processus de vérification dans le bios, cette manière est moins robuste dû aux attaques possibles sur ce genre d'application, ou d'une manière matérielle en intégrant des matériels spécialisés dans les vérifications pour vérifier l'intégrité de l'hyperviseur. Cette manière est très robuste mais coûteuse à cause des besoins matériels.

2.4. Micronoyaux

Byzantine Fault désigne des erreurs arbitraires qui surviennent lors de l'exécution d'un algorithme par un système distribué, il englobe les erreurs dans l'omission (l'envoi ou la réception d'une requête) et dans la commission (traitement incorrect de la requête). Quand une erreur de ce type arrive, le système peut réagir d'une façon imprévisible à moins que le système soit désigné à avoir une tolérance à ce genre de problèmes.

L'utilisation des systèmes tolérants au byzantine fault comme micronoyaux dans les hyperviseurs permet de créer des hyperviseurs qui se protègent contre les attaques. Des hyperviseurs désignés comme des micronoyaux existent déjà, mais l'extension de ces types de ces derniers pour être tolérants au byzantine fault reste un concept théorique pour l'instant[6].

2.5. Protection du couche de l'hyperviseur

Une autre façon pour protéger les hyperviseurs est de protéger la couche dont ils se trouvent. GuardType a présenté ce genre de protection en étant un hyperviseur sur les hyperviseurs permettant seulement l'exécution des hyperviseurs fiables en créant une couche matérielle virtuelle comme illustré dans la figure III-9, le problème de cette approche est le fait qu'elle n'est pas protégée contre les failles des hyperviseurs fiables[6].

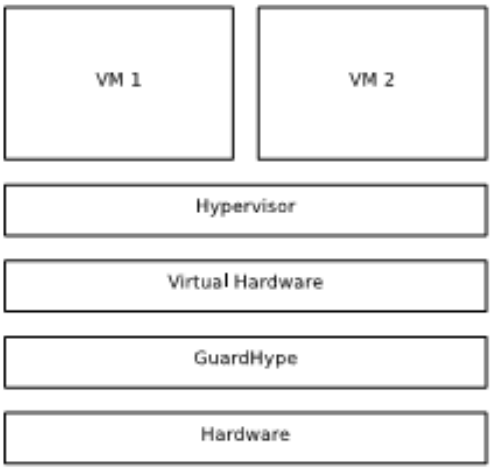


Figure III-9: protection de la couche hyperviseur en utilisant des GuardTypes[6]

2.6. L'intégrité de machines virtuelles

La sécurisation de la virtualisation impose la protection, en plus de l'hyperviseur, des machines virtuelles qui sont lancées au-dessus, ce qui signifie qu'un client doit avoir la possibilité de vérifier l'intégrité de leurs machines virtuelles. En effet cela peut être fait de deux manières[6] :

2.6.1. Vérification logiciel

Cette vérification peut être faite en utilisant des algorithmes de hachages comme SHA lors de lancement de la machine virtuelle pour vérifier son intégrité, le problème de cette approche c'est qu'elle suppose que l'hyperviseur est fiable.

2.6.2. Vérification matériel

Cette vérification utilise principalement un TPM[6] (trusted platforme module) qui permet de garantir que le système reste fiable, l'utilisation de ce genre de module est limitée à une seule machine par module, l'utilisation de ce genre d'approche est presque totalement fiable mais elle demande plus de ressources matérielles.

En 2006 IBM à développer le vTPM [21] une version logiciel de TPM, qui permet lors de son intégration dans l'hyperviseur de profiter de toutes les fonctionnalités de TPM sur un grand nombre de machines virtuelles.

[6] TPM (trusted platforme module) : est une puce spécialisée utilisée pour effectuer des opérations cryptographiques, comme le stockage de clés de chiffrement pour sécuriser l'information qui est habituellement utilisé par le système hôte pour authentifier le matériel.

2.7. L'isolation de ressources entre les machines virtuelles

Jusqu'à récemment, l'isolation se faisait par des moyens logiciels, ce qui a permis à plusieurs vecteurs d'attaques de cibler la virtualisation des ressources, mais l'introduction de l'IOMMU (input/output memory management unit) a permis une gestion meilleure des ressources[6] :

2.7.1. Sans protection matériel

- **Disque** :

L'isolation logicielle a permis aux utilisateurs d'accéder seulement aux zones mémoires désignées à eux, mais cela n'a pas limité les privilèges des hyperviseurs, ce qui fait qu'un hyperviseur peut lire, écrire, modifier, supprimer les informations des autres utilisateurs s'il est contrôlé par un administrateur malveillant.

Ce problème peut être géré en utilisant des algorithmes de cryptographie mais cela garantit seulement la confidentialité des données, il est toujours possible pour un hyperviseur de détruire les données.

- **Réseau :**

Les privilèges de l'hyperviseur lui permettent d'intercepter et de lire tout le trafic passant dans le réseau virtuel, ce qui impose l'utilisation des algorithmes de cryptage pour garantir la confidentialité et des réseaux virtuels privés (VPN) pour garantir l'intégrité des informations transmises par réseau.

Le problème dans cette méthode c'est qu'on utilise des inspections mémoire approfondis, un administrateur a la possibilité de lire toutes les informations concernant le VPN ce qui lui donne la possibilité de se connecter et d'espionner sur ce réseau.

2.7.2. Avec protection matériel

L'utilisation des hyperviseurs supportant le IOMMU permet de déplacer les diapositifs de DMA (Direct Memory Access) à des zones moins privilégiés ce qui permet l'accès seulement à la mémoire occupée par ses pilotes.

Il est toujours possible d'attaquer les cartes réseau mais cela permet seulement d'accéder à des zones mémoires spécifiques dans ces cartes, ce qui fait que si la communication passante par cette carte réseau est chiffrée, l'attaque sur ces cartes devient inutile. La même chose pour les disques, tant que les informations stockées sont chiffrées, l'attaquant n'as pas la possibilité de récupérer les informations.

Le problème de cette approche c'est qu'elle est limitée aux dispositifs DMA et n'est pas appliquée au CPU et aux contrôleurs de mémoire qui doivent être fiable.

3. Sécurité des données

Data Security est l'un des plus grand points faibles de la technologie du Cloud ce qui a poussé plusieurs chercheurs à proposer des solutions spécifiques à chaque étape du cycle de vie de l'information, en effet peu importe l'étape du cycle de vie de l'information l'intégrité et la confidentialité restent les points les plus importants.

3.1. Confidentialité

Différemment au modèles traditionnels dans lesquels la confidentialité peut être assurée par une implémentation d'un algorithme de cryptographie où le propriétaire est le seul qui a la possibilité de déchiffrer les informations, le Cloud pause des contraintes que le propriétaire des informations doit les prendre en considération à moins que s'il veut stocker ses données en stockage passif (il n'y a ni partage ni utilisation online de ces données) et même dans ce cas une couche de chiffrement doit être appliquée.

Le Cloud offre la possibilité de stocker l'information mais sa nature pose un grand risque sur la confidentialité des données parce que le fournisseur de service et le client ne se trouvent pas dans le même domaine, donc il y a une possibilité d'avoir des fuites d'information, soit intentionnel si le fournisseur est malveillant, soit involontairement suite à une attaque, ce qui impose le chiffrement des informations, mais ce chiffrement impose des limitations au cas où le propriétaire veut partager des informations spécifiques avec un client ou même dans le cas de recherche, dans ce qui suit nous allons présenter quelques méthodes permettant le partage des informations tout en conservant la confidentialité et en implémentant le principe de granularité.

3.1.1. Méthode 1 : méthode de base

Le partage des informations peut être fait d'une manière simple en chiffrant les informations et en partageant la clé de chiffrement avec les clients, mais cela va révéler des informations que le propriétaire n'a pas l'intention de les partager, on peut améliorer cette approche en décomposant les informations en groupes de blocs[7] et les chiffrer avec une clé spécifique pour chaque bloc d'information comme illustré dans la figure III-10 puis on partage la clé associée au fichier avec le client qui lui correspond dans les droits d'accès.

[7] Dans ce contexte un bloc veut dire la quantité minimale d'informations qu'on veut partager

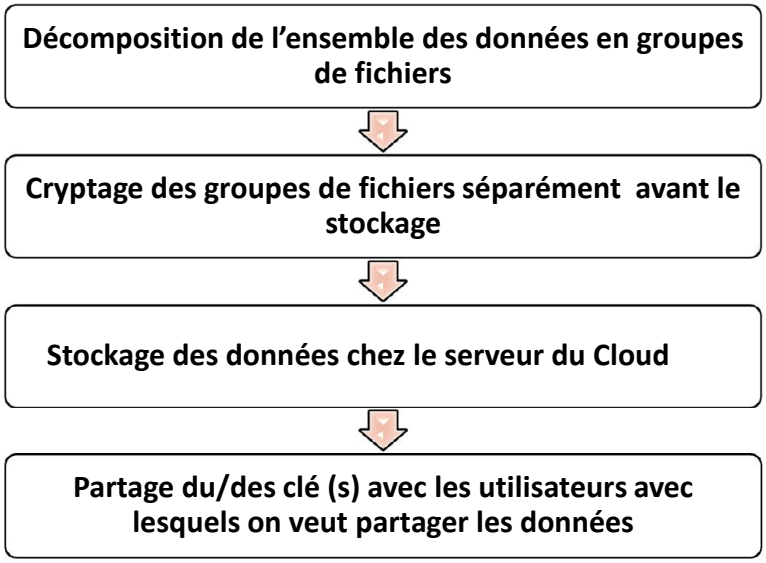


Figure III-10: Méthode basique

Le problème de cette approche c'est que le nombre de clés va augmenter avec le nombre de blocs d'informations construits. En plus le propriétaire doit être toujours online pour faire la tâche de gestion des droits d'accès, ce qui va surcharger sa bande passante dans le processus de distribution des clés et en cas de révocation des droits d'accès d'un client, le propriétaire doit télécharger tous les blocs associés à ce client, les déchiffrer et les rechiffrer avec la nouvelle clé, puis les recharger sur le serveur de stockage, ensuite il est obligé de faire la mise à jour des clés des autres clients, ce qui devient de plus en plus compliqué avec l'accroissement de ces derniers.

3.1.2. Méthode 2: Attribute-based encryption for fine-grained access control of encrypted data

La méthode proposée par Vipul Goyal, Omkant Pandey, AmitSahai et Brent Waters[22] se base sur l'idée d'associer à chaque fichier un ensemble d'attributs et générer des clés en se basant sur ces derniers, cette méthode se compose de 4 étapes illustrées dans la figure III-11:

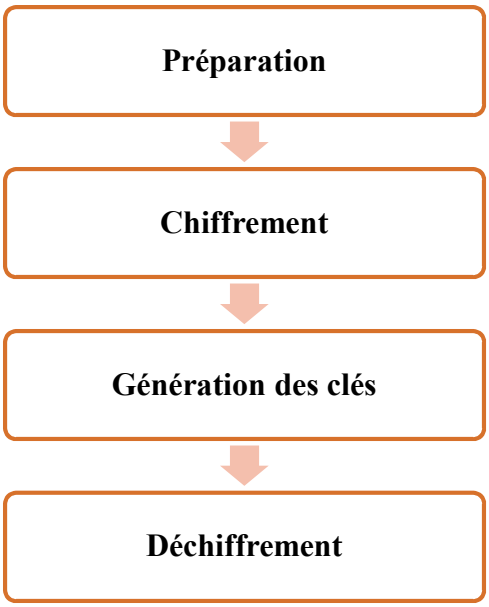


Figure III-11: Méthode 2

Généralement l'idée de cette approche c'est de chiffrer chaque fichier avec une clé basée sur les attributs y associés puis créer des clés privées des clients en se basant sur leur structure d'accès pour leur permettre de déchiffrer seulement les fichiers auxquels ils ont permission (voir la figure III-12) :

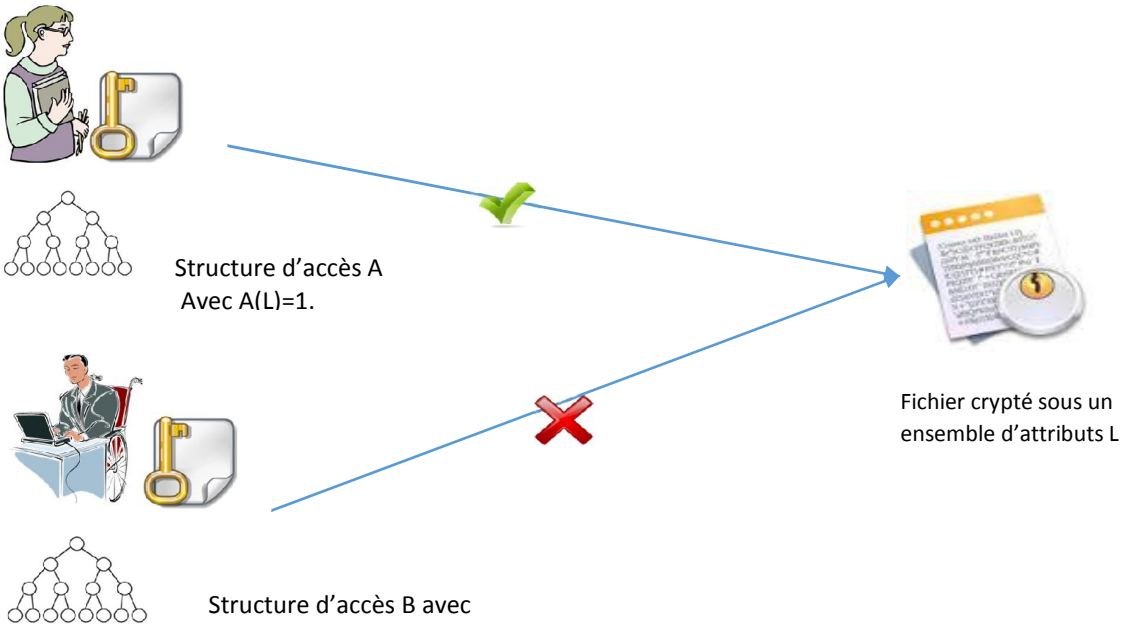


Figure III-12: exemple de l'implémentation de la méthode KP-ABE

Dans cette approche le propriétaire des données associe chaque fichier avec un ensemble des attributs et utilise la méthode KP-KBE (Key Policy Attribut based Encryptions). Cette approche consiste en 4 phases :

- Préparation : dans cette phase un algorithme qui utilise un paramètre de sécurité implicite, génère la clé publique PK et la clé maitre MK.

  Pour générer les clés, on définit un univers des attributs $= \{1,2, \ldots., n\}$, pour chaque attribut $i \in U$ on choisit un nombre t_i uniformément aléatoire sur Z_p et on choisit un nombre y uniformément aléatoire dans Z_p et on construit les clés sous la forme suivante :

$$MK = (y, t_1, t_2, \ldots, t_n)$$
$$PK = (Y, T_1, T_2, \ldots, T_n)$$
$$\text{Avec } T_i = g^{t_i} \text{ et } \quad et\ Y = e(g,g)^y$$

- Chiffrement : dans cette phase un algorithme prend en paramètre la clé publique PK, le message m_0 et un ensemble d'attributs γ et retourne le message chiffré E.

$$E = \{\gamma, E' = m_0 Y^s, \{E_i = T_i^s\}_{i\in\gamma}\}.$$
$$avec\ s\ un\ nombre\ aléatoire\ tel\ que\ s \in Z_p$$

- Génération des clés : pour générer la clé un algorithme prend en paramètre une structure d'accès A, la clé maitre MK et la clé public PK et génère une clé de déchiffrement D, cet algorithme doit générer une clé de déchiffrement qui respecte la condition $T(\gamma) = 1$, ce qui veut dire que la clé ne va fonctionner (décrypter les données) que si tous les attributs associés au fichier de données se trouve dans la structure d'accès du client.

- Déchiffrement : cet algorithme prend en paramètre le message E' qui est chiffré sous un ensemble des attributs γ, la clé de déchiffrement D qui est générée en se basant sur la structure d'accès T et la clé public P et retourne le message déchiffré M.

En effet cette méthode permet le partage des informations tout en gardant la confidentialité des données, mais ce processus impose un coût important en terme de bande passante et de puissance de calcul sur le propriétaire des données qui est le responsable de la gestion des droits d'accès, en plus cette approche n'a pas présenté une méthode de révocation des droits d'accès optimisée, par défaut l'idée basique pour effectuer la révocation des droits d'un client x c'est de remplacer l'ensemble d'attributs γ associé à x tel que $T_x(\gamma) = 0$, ce processus doit être fait par le propriétaire lui-même ce qui n'est pas faisable pour un client qui a des limitations en terme de la bande passante et de la puissance.

3.1.3. Méthode 3: Achieving secure, scalable, and fine-grained data access control in cloud computing

Une autre implémentation de la méthode PK-ABE a été présentée par Yu, Shucheng, Wang, Cong, Ren, Kui, Lou, Wenjing[23]dans le domaine du Cloud, cette implémentation a comme idée de déléguer les opérations, qui représentent un fardeau en terme de calcul sur le propriétaire, au serveur Cloud sans divulguer le contenu des fichiers.

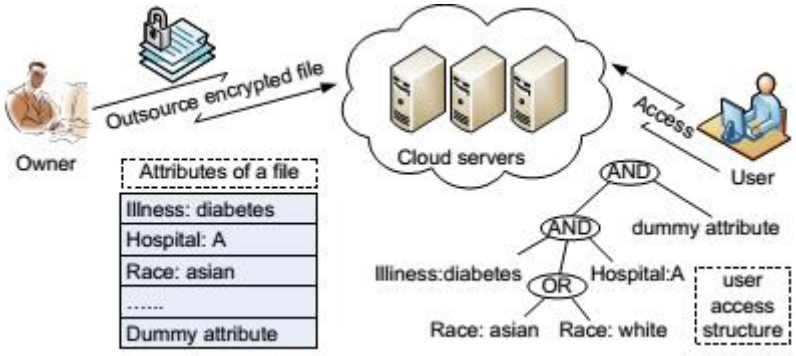


Figure III-13: Exemple de l'implémentation[23]

Similairement à la méthode précédente, le propriétaire des données commence par générer la clé public PK et la clé maitre MK, mais au lieu de les utiliser pour crypter les données ces clés sont utilisées pour crypter la clé de chiffrement symétrique utilisée pour chiffrer un fichier, cette approche se compose de cinq étapes :

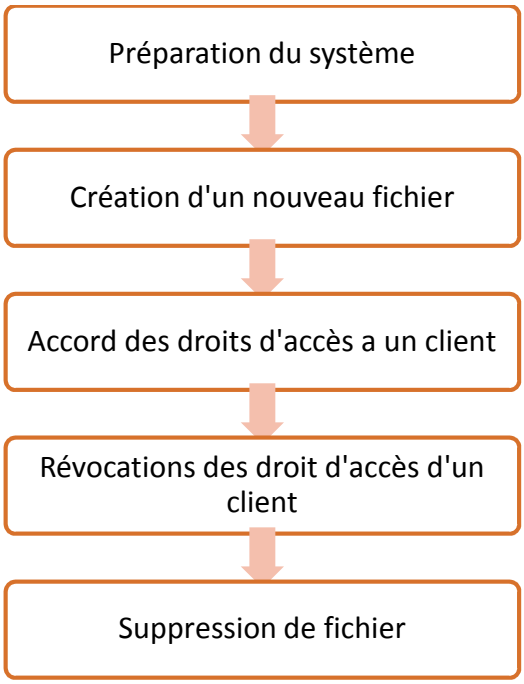
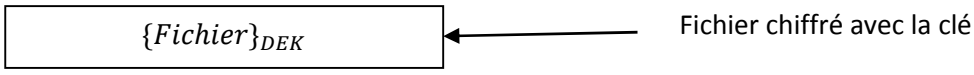


Figure III-14: Méthode 3

- Préparation du système : dans cette phase le propriétaire de fichiers commence par choisir un paramètre de sécurité k et fait l'appelle à la fonction $ASetup(k)$ qui va rendre la clé maitre Mk et la clé public Pk, le propriétaire va signer par la suite les clés générées et envoyer la clé Pk avec les signatures au serveur Cloud.

- Création d'un nouveau fichier : avant de stocker le fichier chez le fournisseur du Cloud, le propriétaire du fichier doit le préparer :

Premièrement, il commence par attribuer un identifiant unique ID au fichier, puis il choisit une clé de chiffrement symétrique DEK et chiffre le fichier avec.



Après le chiffrement du fichier, le propriétaire procède vers la définition d'un ensemble d'attributs I associé au fichier puis chiffre la clé DEK avec I en utilisant la méthode KP-ABE :

$$AEncrypt(I, DEK, PK) \rightarrow (\tilde{E}, \{E_i\}_{i \in I})$$

Enfin, il concatène la clé chiffrée avec les données cryptées et l'identifiant de fichier avant de les envoyer vers le serveur Cloud.

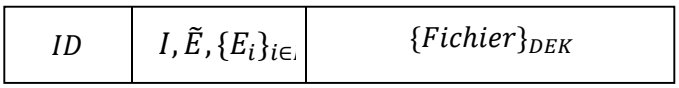


Figure III-15: Format du fichier stocké chez le fournisseur du Cloud

- Accord des droits d'accès à un client : lors de l'ajout d'un nouveau client au système, le propriétaire doit lui accorder une structure d'accès et une clé privée comme suit :

Le propriétaire commence par attribuer une identité unique w et une structure d'accès P au client, puis il génère la clé privée du client en se basant sur cette structure d'accès et la clé maitre du propriétaire Mk.

$$SK \leftarrow Akeygen(P, MK)$$

La clé du client avec sa structure d'accès, la clé publique du propriétaire et la signature du propriétaire de ces derniers ($P, SK, PK,$ δ□$O.\ P, SK, PK$□□) seront par suite chiffrées avec la clé publique du client produisant C.

Le propriétaire envoie$C$ avec la liste de composantes de la clé privée du client SK et leur signature $(T, C, \delta_{O.(T,C)})$ vers le serveur cloud, T désigne la liste des composantes associées aux attributs à part les attributs Att_D fictifs $(w, \{j, sk_j\}_{j \in L_p \setminus Att_D})$

de cette façon, même si le serveur stocke ces composantes, il ne peut pas recréer la clé privée du client, de cette façon on peut mettre à jour les droits d'accès en cas de révocation sans mettre en risque le contenu du fichier.

Lors de la réception, le serveur vérifie la signature$\delta_{O.(T,C)}$, si la signature est authentique il procède à envoyer C au client et à enregistrer T dans la liste d'utilisateurs UL.

- Révocation des droits d'accès : intuitivement la révocation des droits d'accès peut être réalisée par le propriétaire en calculant le nombre minimum des attributs à modifier à fin de révoquer les droits d'un client, puis modifier leur composantes dans $MK\ et\ PK$ puis modifier ces attributs dans les composantes de SK de chaque client et re-chiffrer les DEK avec les nouvelles données.

Le problème dans ce processus c'est qu'il impose un grand fardeau en termes de calcul et impose que le propriétaire doit rester online pour faire cette gestion. Pour remédier à ce problème les auteurs ont implémenté une méthode qui vise à déléguer le processus de re-chiffrement au Cloud en utilisant une technique de proxy-encryptions[8], de cette façon le propriétaire n'a qu'à modifier les informations concernant le $PK\ et\ MK$ puis envoyer les attributs modifiés dans PK avec la liste des attributs sélectionnés vers le serveur qui va faire le re-chiffrement et la mise à jour des clés des clients.

3.1.4. Méthode 4: Over-encryption: management of access control evolution on outsourced data

Une approche qui propose l'exploitation de la notion de dérivation des clés a été proposée par Di Vimercati ,Sabrina De Capitani, Foresti Sara ,Jajodia Sushil, Paraboschi Stefano et Samarati, Pierangela [24], l'idée basique de cette approche consiste en trois étapes :

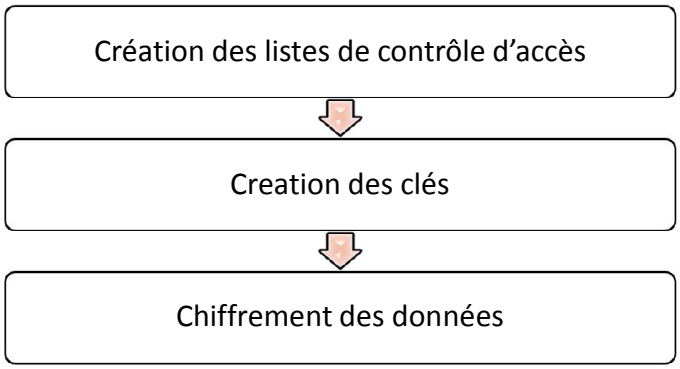


Figure III-16: approche à une seule couche

[8] Proxy-encryptions ; c'est une méthode permettant à un agent intermédiaire de rechiffrer les données cryptées par un individu A afin qu'elles soient décryptées par un individu B

Pour illustrer la méthode présentée par les auteurs on se basera sur l'exemple suivant, on supposera le cas dans lequel on a cinq clients (A,B,C,D,E) et huit ressources à partager (r1,r2,r3,r5,r5,r6,r,7,r8) avec la structure d'accès suivante :

$$A \rightarrow r5, r6, r7, r8$$
$$B \rightarrow r5, r6, r7, r8$$
$$C \rightarrow r1, r2, r3, r4, r5, r6, r7, r8$$
$$D \rightarrow r3, r4$$
$$E \rightarrow r8$$

Le but de cette approche c'est de proposer une méthode dans laquelle on va partager les données tout en minimisant le nombre des clés stockées par les clients pour réaliser cela on doit définir une structure d'accès de puis la matrice du contrôle « tableau III-1 », La méthode directe c'est d'exploiter la hiérarchie entre les groupes des clients pour et d'en tirer une hiérarchie des clés.

Tableau III-1: Matrice de contrôle d'accès

	R_1	**R_2**	**R_3**	**R_4**	**R_5**	**R_6**	**R_7**	**R_8**
A	0	0	0	0	1	1	1	1
B	0	0	0	0	1	1	1	1
C	1	1	1	1	1	1	1	1
D	0	0	1	1	0	0	0	0
E	0	0	0	0	0	0	0	1

D'après la matrice des contrôles d'accès ci-dessous « tableau III-1 » on peut grouper les utilisateurs sous quatre groupes différents selon les ressources auquel ils ont accès {C, ABC, CD, ABCE} ce qui fait on est amené à créer huit différentes clés ; cinq clés associées aux individus {A, B, C, D, E} et les clés associées au groupes {ABC, CD, ABCE}, puis on va crypter les données avec la clé appropriée.

Généralement les clés associées à chaque groupe de contrôle d'accès est dérivable à partir de la clé de chaque individu et un attribut associé à ce dernier, cette relation peut être illustrée en format de graphe « figure III-17 » avec v_i représente la clé i :

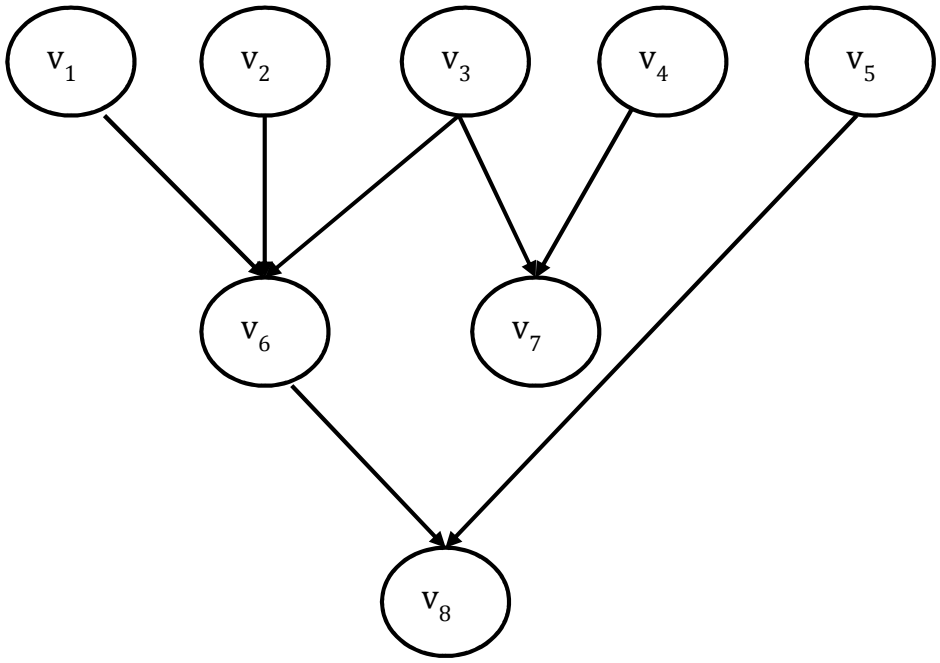


Figure III-17: hiérarchie de dérivation des clés

Chaque clé est associée à son prédécesseur par un jeton défini par la fonction $t_{i,j} = k_j \oplus \mathrm{h}(k_i, l_j)$ avec $\oplus$ une fonction XOR, l_j une étiquette publique associée à la clé $k_j$et h une fonction cryptographique déterministe[25] ce qui rend possible la définition d'un ensemble de relations entre les différentes clés comme représenté dans le tableau III-2 :

Tableau III-2 : relations entre les clés

Source	*Destination*	*Valeur*
$\boldsymbol{v1.k}$	$v6.k$	$v_6.k_6 \oplus \mathrm{h}(v_1.k_1, l_6)$
$\boldsymbol{v2.k}$	$v6.k$	$v_6.k_6 \oplus \mathrm{h}(v_2.k_2, l_6)$
$\boldsymbol{v3.k}$	$v6.k$	$v_6.k_6 \oplus \mathrm{h}(v_3.k_3, l_6)$
$\boldsymbol{v3.k}$	$v7.k$	$v_7.k_7 \oplus \mathrm{h}(v_3.k_3, l_7)$
$\boldsymbol{v4.k}$	$v7.k$	$v_7.k_7 \oplus \mathrm{h}(v_4.k_4, l_7)$
$\boldsymbol{v5.k}$	$v8.k$	$v_8.k_8 \oplus \mathrm{h}(v_5.k_5, l_8)$
$\boldsymbol{v6.k}$	$v8.k$	$v_8.k_8 \oplus \mathrm{h}(v_6.k_6, l_8)$

$v_i.k$: *laclé kassocié alasommet*v_i

Cette méthode limite le nombre de clés gérées par les clients et permet la gestion hiérarchique des clés tout en conservant la confidentialité des données, en plus elle permet de diminuer le nombre de clés utilisées en utilisant seulement les clés associées au groupes de

contrôle d'accès (qui représente un ou plusieurs client(s)) comme illustré dans le tableau III-4, l'application de cette méthode sur notre exemple a permis de limiter le nombre de clés utilisées à quatre tout en gardant le principe de granularité, de partage et de confidentialité des données.

Tableau III-3: Clé associée à chaque groupe de contrôle d'accès

Groupe d'accès	Clé associé
A	$v1.k$
B	$v2.k$
C	$v3.k$
D	$v4.k$
E	$v5.k$
{A B C}	$v6.k$
[2]	$v7.k$
{A B C E}	$v8.k$

Tableau III-4: clé utilisée pour chiffrer chaque ressource

Ressources	Clé Utilisé
r_1, r_2	$v3.k$
r_3, r_4	$v7.k$
r_5, r_6, r_7	$v6.k$
r_8	$v8.k$

Le problème dans l'utilisation de l'approche directe c'est qu'elle est adaptée à une structure d'accès fixe, cela veut dire que le propriétaire doit prédéfinir la structure d'accès et préparer les clés avant l'envoie des ressources au serveur et une fois que les ressources sont envoyées il devient difficile et très coûteux pour le propriétaire de modifier la structure, pour cette raison les auteurs ont proposé une approche à deux couches « BEL » (Basic Encryptions Layer) et « SEL » (Surface Encryptions Layer).

- **Basic Encryptions Layer** : cette couche de chiffrement est performée avec la structure d'accès définie à la phase initiale, la différence entre cette couche et la méthode directe c'est que dans « BEL » on distingue deux types de clé :
  - Clé de dérivation : cette clé est associée à chaque utilisateur et permet la dérivation des clés avec la fonction de hachage.
  - Clé d'accès : c'est la clé utilisée pour chiffrer les ressources, cette clé peut être trouvée par l'application d'une fonction de hachage sur la clé de la dérivation $k_a = h(k)$.

- **Surface Encryptions Layer :** cette couche est effectuée par le serveur sur les données envoyées par le client, en utilisant des clés *k* associées, d'une façon similaire au modèle directe, à chaque nœud de l'arbre tel que les clés associées au prédécesseur d'un nœud peut être trouvées à partir de la clé associée à cette nœud en utilisant une fonction de hachage.

La combinaison de ces deux couches peut être faite de deux manières :

- Delta-SEL : dans cette combinaison les données sont chiffrées seulement en utilisant BEL, l'utilisation du chiffrement de la couche SEL n'est établie que lors du changement des droits d'accès.
- Full-SEL : dans cette combinaison les données sont chiffrées à la fois avec la couche BEL et la couche SEL ce qui permet de renforcer le contrôle des droits d'accès et de maximiser la sécurité mais cela impose la nécessité de double chiffrement ce qui pose un coût supplémentaire au niveau de calcul.

Dans tous les cas, l'accès à un ressource qui est protégé par les couches BEL et SEL nécessite au client d'avoir les clés pour les deux niveaux, ce qui rend les opérations de mise à jours délicates (on doit mettre à jour les droits d'accès d'une façon que le nombre de clé reste minime et que les droits d'accès doivent être respectés), on distingue quatre opérations de mise à jour (attribution, révocation).

Attribution des droits d'accès : dans le cas d'attribution des droits d'accès on commence par vérifier s'il existe un groupe d'accès qui a déjà les mêmes droits que ceux du client mise à jour si c'est le cas on dérive la clé du groupe à partir de sa clé et on stocke le jeton chez le fournisseur, si ce n'est pas le cas on doit créer un nouveau nœud et générer un nouveau mot de passe puis mettre à jour la base de données des jetons tel qu'elle prend en considération les nouveaux droits sans donner l'accès à des ressources non prise en charge.

Exemple :

On va attribuer au client E l'accès à la ressource r4 (Grant(E, r4)), on commence tout d'abords par mettre à jour la matrice de contrôle d'accès :

Tableau III-5: nouvelle matrice de contrôle d'accès

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
A	0	0	0	0	1	1	1	1
B	0	0	0	0	1	1	1	1
C	1	1	1	1	1	1	1	1
D	0	0	1	1	0	0	0	0
E	0	0	0	**1**	0	0	0	1

Après la mise à jour de la liste de contrôle d'accès on entame la recherche d'un groupe de contrôle qui a les même droits que notre client ce qui n'existe pas dans notre exemple dans ce cas on crée un nouveau nœud $v9$ et on y génère une clé $v9.k$ puis on chiffre la ressource r4.

Tableau III-6: les clés associées aux ressources

Ressources	Clé Utilisé
r_1, r_2	$v3.k$
r_3	$v7.k$
r_5, r_6, r_7	$v6.k$
r_8	$v8.k$
r_4	$v9.k$

Avec cette mise à jour notre structure des droits d'accès devient :

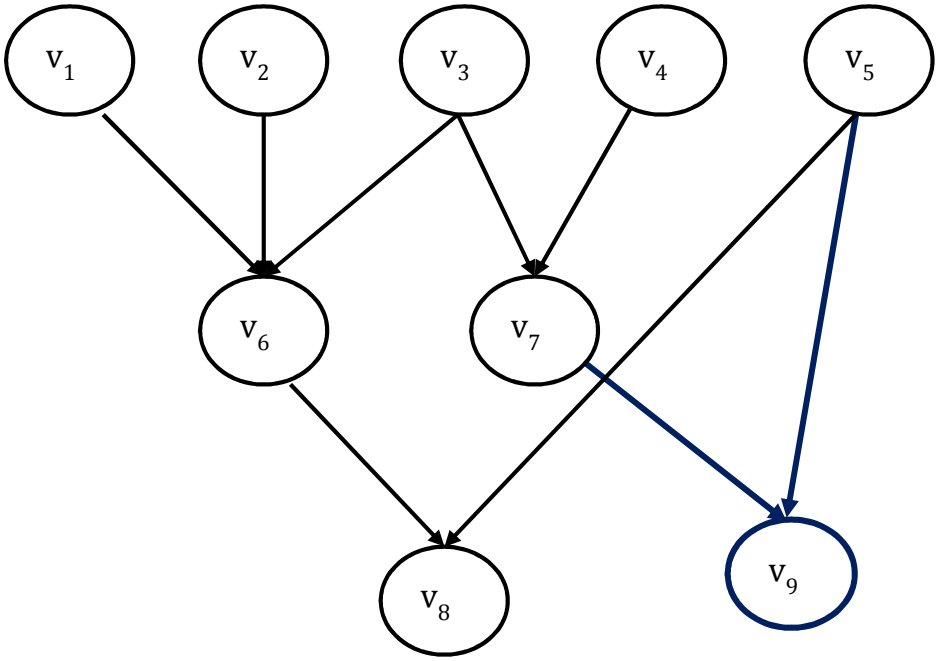


Figure III-18: structure de droits d'accès après l'opération Grant (r4, E)

Enfin des jetons vont être générés pour permettre la dérivation de la clé $v9.k$ à partir de la clé $v7.k$ et $v5.k$.

- Révocation des droits d'accès : le cas de révocation des droit est similaire à celui de l'attribution, on commence par vérifier s'il existe un groupe d'accès qui a déjà les mêmes droits que ceux du clients associés au ressource mise à jour, si c'est le cas on utilise la clé associée au groupe pour rechiffrer les données , si ce n'est pas le cas on doit créer un nouveau nœud et générer un nouveau mot de passe puis mettre à jour la base de données des jeton tel qu'elle prend en considération les nouveaux droits sans donner l'accès à des ressources non prises en charge.

  Exemple : Revoque (E,r8)

  On va révoquer les droits d'accès du client E au ressource r8 (revoque(E,r8)), on commence tout d'abords par mettre à jour la matrice de contrôle d'accès :

Tableau III-7:Nouvelle matrice de contrôle d'accès

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
A	0	0	0	0	1	1	1	1
B	0	0	0	0	1	1	1	1
C	1	1	1	1	1	1	1	1
D	0	0	1	1	0	0	0	0
E	0	0	0	0	0	0	0	**0**

Après la révocation, la ressource r8 se trouve dans le même groupe que les ressources r5, r6, r7 ce qui signifie que la ressource r8 sera rechiffrée en utilisant la clé V6.k :

Tableau III-8: les clés associées aux ressources après l'opération révoque (r8, E)

Ressources	Clé Utilisé
r_1, r_2	$v3.k$
r_3	$v7.k$
r_5, r_6, r_7, r_8	$v6.k$
r_4	$v9.k$

Donc notre nouvelle structure d'accès devient :

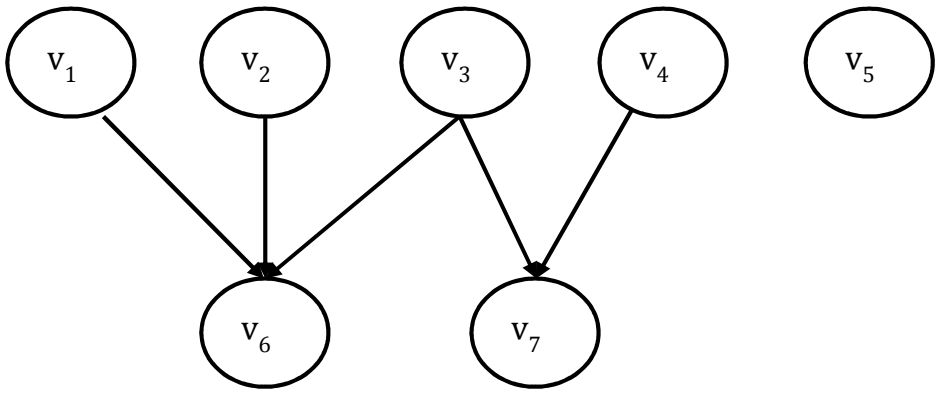


Figure III-19: Structure de droits d'accès après l'opération révoque (r8, E)

L'approche des auteurs permet le partage des ressources tout en assurant la granularité et la confidentialité, mais la complexité de cette approche s'accroit très vite avec le nombre de ressource et de client ce qui rend la gestion plus en plus compliquée.

3.1.5. Méthode 5 :Secure and efficient access to outsourced data

Wang, W. et al. ont proposé une approche[26] permettant l'accès sécurisé au données tout en permettant l'extensibilité et la gestion des droits d'accès, leur approche se caractérise par la communication entre le client, le propriétaire et le serveur, ce modèle de communication est illustré dans la figure III-20 :

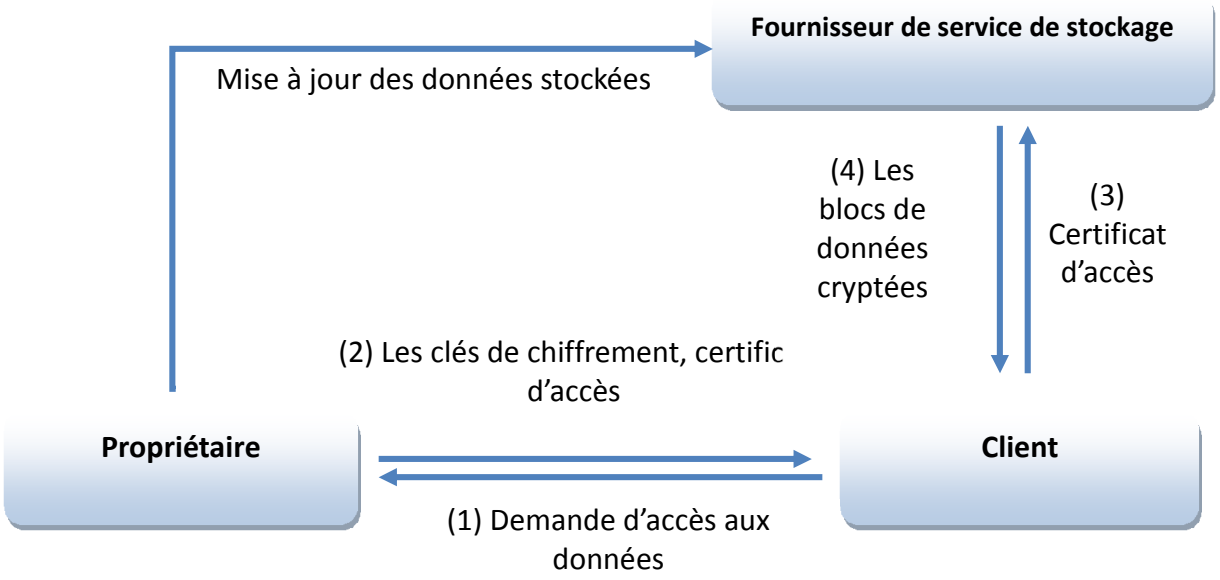
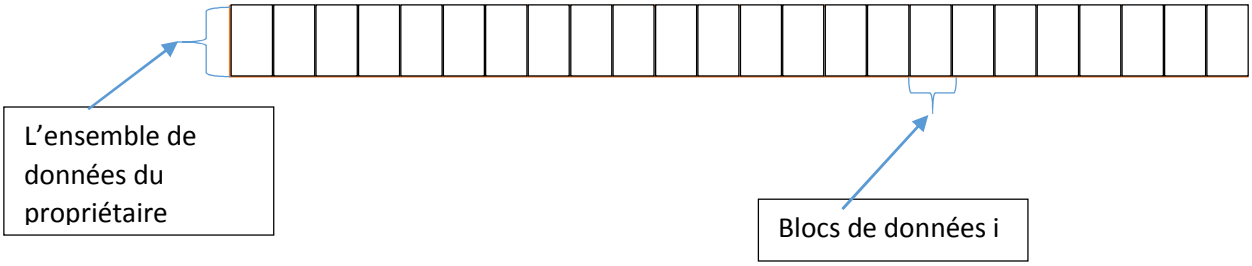


Figure III-20: model de communication

Toutes les tâches de gestion sont gérées par le propriétaire qui va préparer ces données avant de les envoyer au serveur d'une façon que le nombre de clés gérées par lui-même ou par les clients soit limité en une seule clé, pour permettre cette limitation le propriétaire doit chiffrer les données de la façon suivante :

- Le propriétaire des données commence par fragmenter les données sous format d'un ensemble de blocs[9] pour assurer la granularité de partage :

[9] Dans ce contexte un bloc veut dire la quantité minimale d'informations qu'on veut partager



- Le propriétaire doit chiffrer chaque bloc avec une clé séparément, ce qui implique la nécessité de créern clés différentes pour n blocs, cette génération des clés va générer un fardeau si elle est gérée d'une manière traditionnelle dans laquelle chaque clé est indépendante de l'autre, cela va causer une utilisation importante de bande passante lors du communication et chaque client doit gérer un nombre important des clés ce qui a poussé les auteurs à implémenter une méthode de dérivation des clés [26] qui permet la dérivation de clés à partir d'une clé parent et des données publiques, ce qui permet de générer les clés sous forme hiérarchique (figure III-21) :

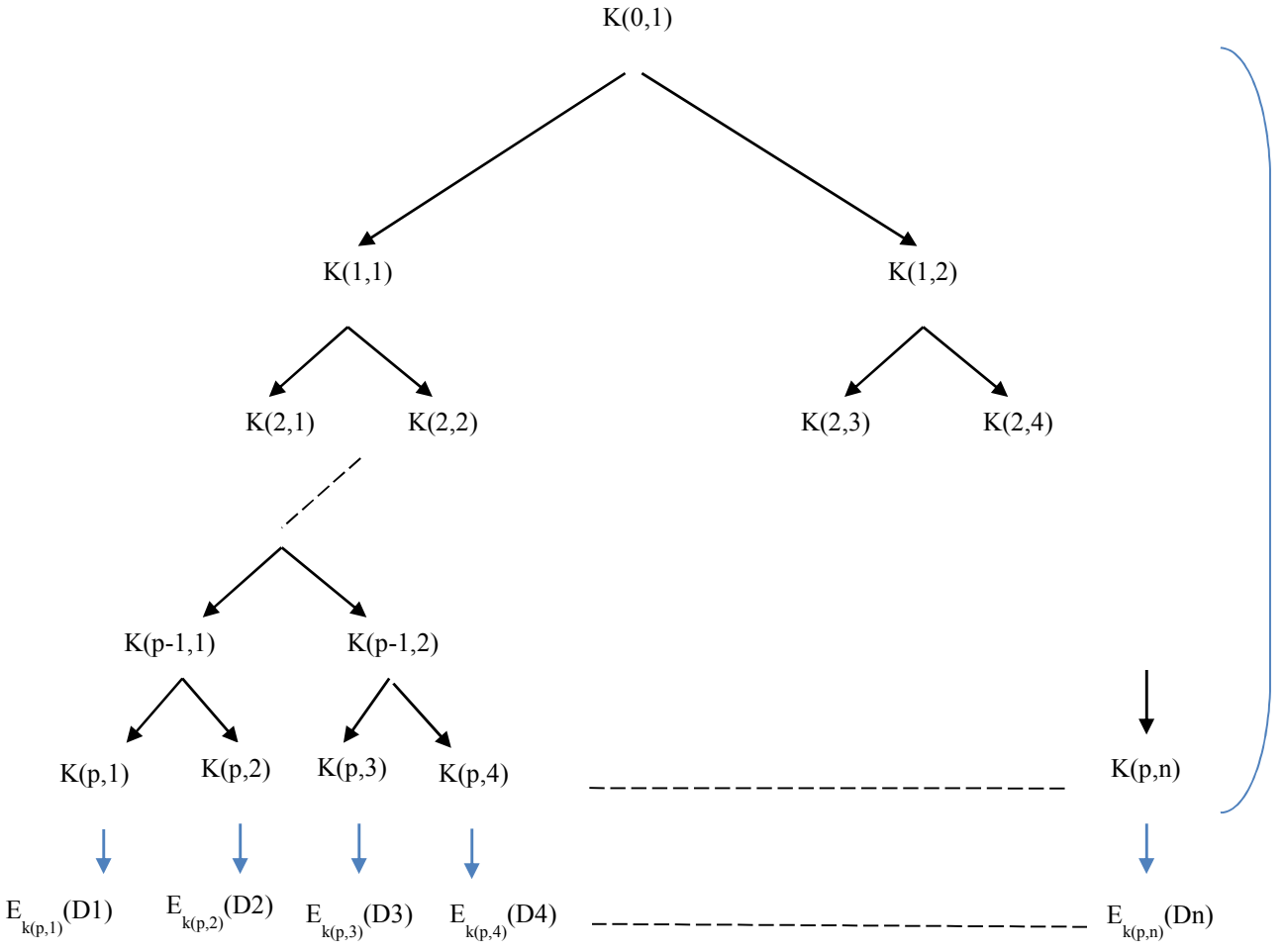


Figure III-21: la hiérarchie de dérivation des clés

Chaque bloc est par suite chiffré avec une clé, puis le propriétaire des données envoie l'ensemble des blocs avec les informations permettant la dérivation vers le serveur

- Pour permettre l'accès à une ressource x chiffrée par K(2,4) et une ressource y chiffrée par K(2,3) le propriétaire envoie la clé K(1,2) au client qui va par suite l'utiliser pour dériver les clés associées aux fichiers auxquels il veut accéder, en effet le propriétaire doit envoyer seulement les clés permettant l'accès aux ressources appropriées ; dans le cas d'un client qui a l'accès seulement à une ressource cryptée par K(p-1,1) le propriétaire doit envoyer seulement la clé K(p-1,1) non pas K(p-2,1) pour ne pas donner des droits non appropriés au client.

L'accès aux données suit la procédure décrite[10]dans la figure III-21, l'approche suppose que le propriétaire partage avec le client une clé secrète KOU et avec le serveur une clé secrète et la clé partagée entre O et S, la procédure d'accès suit les étapes suivantes :

- Demande d'accès aux données : le client demande l'accès d'une ressource de la part du propriétaire des données en lui envoyant un message du format suivant :

U → O: {U, O, E_{kOU} (U, O, request index, data block indexes, MAC code)}

- Le propriétaire identifie le client avec un secret partagé entre eux et il va utiliser « request index », qui va être incrémenté à la fin de chaque demande, pour se protéger contre les attaques à rejeux, puis il va vérifier que les indexes demandés se trouve dans les droits d'accès du client, si c'est le cas le propriétaire va identifier le nombre minimal de clés permettant de couvrir les index demandés sans déplacer les droits attribués et génère un message de réponse de la forme suivante :

O → U: {O, U, request index, ACM index, seed for P(), K', cert for S, MAC code)}

- Suite à la réception du message de la part du serveur le client va par suite extraire le certificat fourni par le propriétaire des données « cert » qui est de la forme suivante :

[10] Abréviation :
- O : le propriétaire
- S : le fournisseur de service (service provider)
- U : l'utilisateur final (end user)
- K_{OU} : la clé partagée entre O et U
- K_{OS} : la clé partagée entre O et S
- MAC code : The Message Authentication Code
- ACM : Access Control Matrix
- K' : clé permettant la dérivation des clés qui permettent le déchiffrement des blocs de données demandées
- seed : un nombre aléatoire pour initialiser P ()
- P() : les fournisseurs de services et les utilisateurs finaux partagent un pseudo bit aléatoire de générateur de séquence P ()

{EkOS(U, request index, ACM index, seed, indexes of data blocks, MAC code)}

U→ S: {O, U, request index, cert for S}

- Le serveur compare index de la matrice du contrôle d'accès (ACM index) avec index qu'il possède, si l'index dans le message est inférieur à l'index que le serveur possède, il notifie le client qui va demander un autre certificat, si non le serveur va utiliser « seed » pour initialiser la fonction P() et génère une séquence de bits pour sur-chiffrer les données en appliquant la fonction XOR, puis envoie les données au client.
- Le client va utiliser « seed » pour régénérer la séquence que le serveur a utilisée et l'utilise pour retrouver les blocs d'informations.

L'attribution des droits d'accès dans cette approche est simple, le propriétaire mis à jour l'ACM et notifie le serveur des mises à jour, lors de demande d'accès le client va suivre le processus normal de l'accès aux données.

La révocation des droits d'accès n'est pas si simple que l'attribution, dans le cas de la révocation, le propriétaire met à jour l'ACM et notifie le serveur comme le cas de l'attribution des droits, la différence c'est que la sécurisation en se basant sur ACM tout seul n'est pas suffisant, une couche de sur-chiffrage est nécessaire ce qui n'est pas fournie par tous les fournisseurs de services.

Cette approche fournit une méthode de partage de données avec respect du principe de confidentialité, mais la majorité des tâches de gestion implique le propriétaire ce qui impose sur lui d'être toujours online et impose un fardeau sur ses ressources et la taille de données stockées dépend du type de chiffrage, le type d'algorithme utilisé et le nombre de blocks (tailles des informations publiques).

Méthode	Utilisation des ressources	Extensibilité / contrôle	Granularité de partage	Contrôle d'accès dynamiques	Respecte le principe de confidentialité
Méthode 1 : basique	Elevé	Non	Oui	Non	Oui
Méthode 2: Attribute-based encryption for fine-grained access control of encrypted data)	Importante (toutes les tâches de la gestion et de chiffrement sont effectuées par l'utilisateur)	Oui, l'accès est contrôlé par l'ensemble des attributs définis pour chaque fichier)	Oui, la granularité est assurée par les attributs, on ne partage que les fichiers auxquels l'utilisateur à la clé associée à ces attributs	Oui mais coûteux, le propriétaire doit télécharger et modifier tous les attributs en cas de révocations	Oui
Méthode 3 (Achieving secure, scalable, and fine-grained data access control in cloud computing)	Moyen, le propriétaire ne souffre pas de coût au niveau de mise à jour dans le cas de l'utilisation de « proxy-encryptions » mais il doit être online pour faire la tâche de la gestion et de la mise à jour des attributs	Oui, l'accès est contrôlé par l'ensemble des attributs définis pour chaque fichier		Oui et la tâche est déléguée au serveur ce qui fait l'utilisateur n'a aucun coût relié à la mise à jour des droits d'accès	Oui, même si le serveur garde la majorité des composantes des clés associées au client, le propriétaire prévoie des attributs fictifs que seul le client va avoir ses clés et même dans le cas de modification des droits d'accès ces attributs ne sont pas changés.

Méthode 4 : Over-encryption: management of access control evolution on outsourced data	Moyen, le propriétaire doit faire la tâche de chiffrement initialement et dans le cas de mise à jour de la couche BEL ou l'utilisation de la méthode directe	Oui, les doits sont extensibles mais la complexité de la gestion augmente avec le nombre de clients et le nombre de ressources	Oui, chaque fichier est chiffré avec la clé associée au groupe qu'y ont accès ce qui permet d'ajouter les droits en dérivant la clé ou en re-chiffrant les ressources et la révocation se fait par rechiffrement des ressources	Oui et en cas d'utilisation de couche SEL le coût révocation des droits d'accès devient moins coûteux	Oui, les données sont protégées par une couche de chiffrement avant l'envoi vers le serveur
Méthode 5 :Secure and efficient access to outsourced data	Moyen, même si l'utilisation de ressource est très faible mais chaque opération nécessite que le propriétaire soit online ce qui n'est pas valable à tout le monde	Oui, l'extensibilité est assurée par la matrice de contrôle d'accès, l'ajout de nouveau droit se fait par une simple mise à jour de la matrice de contrôle d'accès	Oui, la granularité est assurée par la matrice de contrôle d'accès et les certificats fournis par le propriétaire	Oui, l'attribution des droits d'accès se fait avec un changement des droits dans la matrice des droits d'accès	Oui, dans le cas d'un serveur qui fait la sur chiffrage les données sont protégées dans le cas de révocation, si non les données sont partiellement protégées en divulguant seulement les données non mises à jour aux quelles l'utilisateur avait accès

3.2. Intégrité

Comme mentionné précédemment, les méthodes de vérification d'intégrité traditionnelles ne sont pas applicables au Cloud à cause du besoin de télécharger les données à chaque vérification, ce qui impose une grande utilisation de la bande passante, gaspillage important du temps, du coût imposé sur la bande passante (facturé par certains fournisseurs comme Amazone) et des calculs, ce qui a imposé la recherche de nouvelles solutions.

3.2.1. Méthode 1 : Méthode de base 1

La solution qui vient automatiquement à l'esprit c'est de décomposer la source des données en plusieurs fragments, pour chaque fragment on calcule initialement un code MAC (Message Authentification Code), pour chaque tentative de vérification, le client reçoit un nombre aléatoire de fragments de la part du fournisseur de service puis il calcule leurs MAC et les compare.

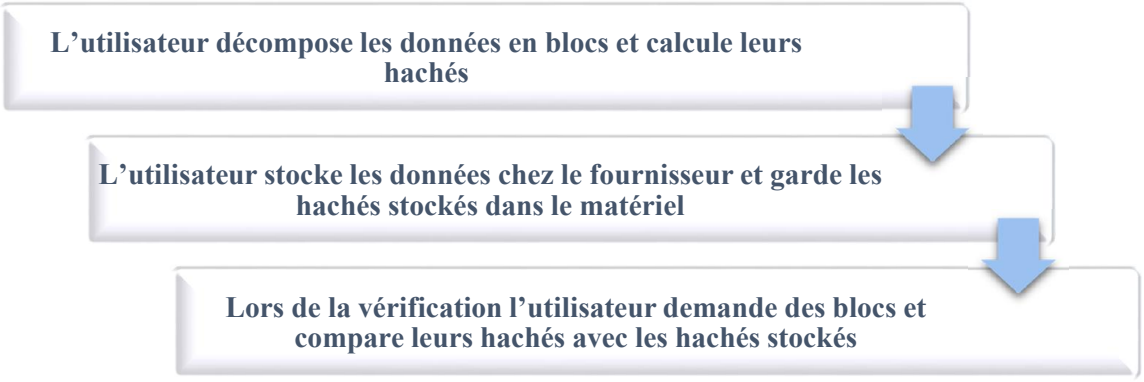


Figure III-22: les étapes de la méthode de base 1

Cette méthode n'est pas vraiment pratique et cela est dû à la charge importante imposée sur le client avant et en cours de la vérification des données :

- Toutes les tâches de cette méthode, de la fragmentation jusqu'à la vérification, sont assurées par le client, ce qui permet – grâce à l'hétérogénéité fournie par le Cloud – d'utiliser des appareils à puissance de calcul faible et à bande passante limitée comme les téléphones mobiles qui ne peuvent pas toujours fournir les ressources nécessaires pour la vérification.
- Pour la vérification, le client est obligé de stocker toutes les signatures (les hachés) des blocs des données qui vont permettre la comparaison lors de la réception des données de la part du serveur, ce qui impose un coût de stockage linéaire avec la taille des données et avec le nombre de fragments prédéterminés (le nombre de hachés stockés).
- Le nombre de vérifications est limité au nombre de hachés prédéterminés, ce qui impose leur recyclage lors de l'épuisement des hachés recalculés, tout en imposant des coûts élevés de calcul et de bande de passante sur le client.

3.2.2. Méthode 2 : Méthode de base 2

On peut améliorer la solution précédente en diminuant le coût imposé sur le client, en déléguant la tâche de vérification à un serveur tierce (comme illustré dans la figure III-23) qui va demander les fragments de la part du serveur et faire la tâche de la vérification.

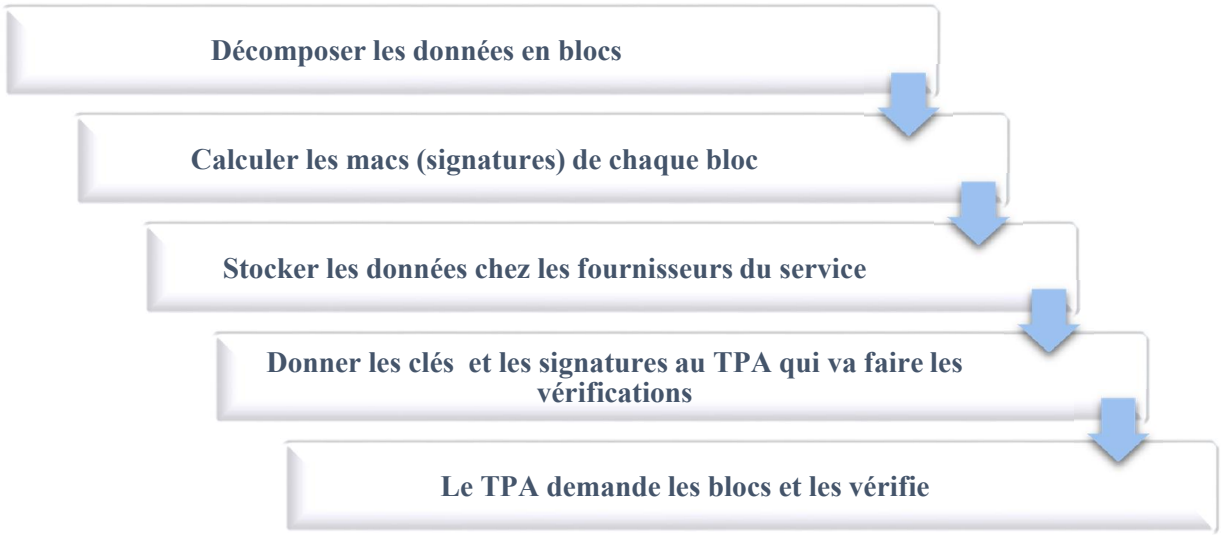


Figure III-23 : Méthode de base 2

Cette méthode permet de diminuer les fardeaux sur le client et même de le libérer du cycle de la vérification (il n'est pas obligé de revérifier chaque période du temps car cette tâche est assurée par le serveur), mais la vérification nécessite l'envoie des données au serveur tierce pour faire la vérification, de cette façon la confidentialité n'est plus respectée. En plus, il y'a toujours un coût lié au transfert des données même si ce coût n'est pas imposé sur le client mais les vérifications périodiques peuvent perturber la qualité des services fournies par le fournisseur de service Cloud lui-même.

Il est possible d'améliorer cette solution en fixant un nombre de vérifications et pré-calculant des macs hachés avec ces dernières et à chaque vérification le client ou le serveur de vérification tierce n'a qu'à demander la vérification en envoyant une clé au serveur et en comparant le mac haché avec le mac pré-calculé. Cependant, le problème dans cette solution c'est que le nombre de vérifications possibles est fortement relié au nombre de signatures prédéfinies, donc à chaque fois qu'on épuise toutes les vérifications possibles, on est amené à télécharger les données puis redéfinir des signatures et aussi à recharger les données dans le serveur, ce qui n'est pas vraiment pratique. En plus, dans le cas d'un serveur malveillant, il est moins coûteux de pré-calculer toutes les signatures et de les stocker que de stocker l'ensemble des données car la taille de ces signatures est très limitée par rapport à la taille des données.

3.2.3. Méthode 3 : Data Integrity Proofs in Cloud Storage

Une autre solution a été présentée par Cong Wang, Qian Wang, Kui Ren et Wenjing Lou[27], elle a comme principe de prendre des données à partir des données stockées sous forme de métadonnées et de les utiliser comme référence. Cette méthode consiste en cinq étapes (comme illustré dans la figure ci-dessous) :

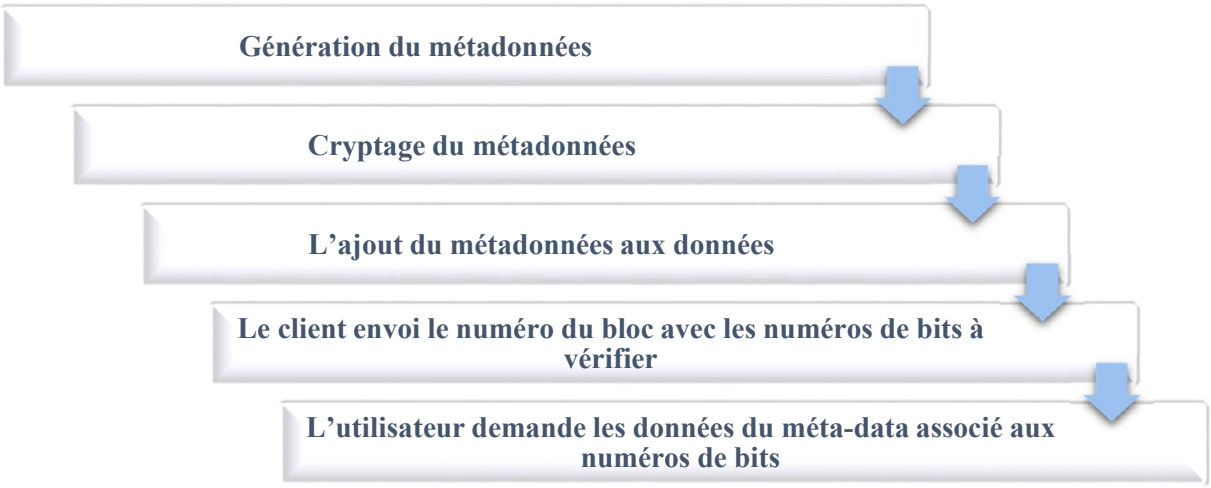
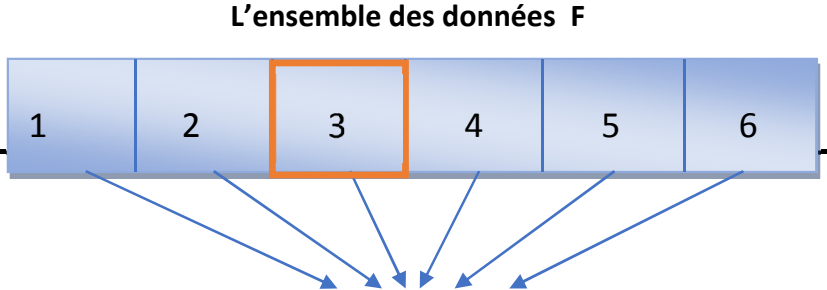


Figure III-24: Les étapes de POR

- Génération des métadonnées :

Dans cette phase, comme illustré dans la figure III-25, le propriétaire des données doit fragmenter les données sous forme de plusieurs fragments, à partir de chaque fragment un nombre *k* de bits prédéterminé par le propriétaire est sélectionné, l'ensemble de bits choisis forment les métadonnées.



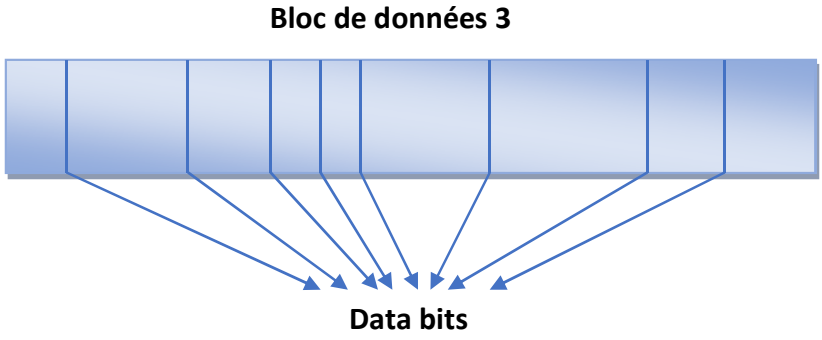


Figure III-25: Génération des métadonnées

- Chiffrement des métadonnées :

Après la sélection d'un ensemble de bits m_id'un bloc de données, ces bits sont chiffrés en utilisant un algorithme de chiffrement pour générer un bloc de métadonnées M_i. Prenons par exemple la fonction XOR :

L'utilisateur va choisir une fonction h permettant de générer un ensemble de k bits α_i, cette fonction doit être connue seulement par le vérifieur :

$$h : i \rightarrow \alpha_i, \qquad \alpha_i \in \{0..2^n\}$$

Chaque bloc M_i de métadonnées est le résultat de l'addition de α_i et m_i :

$$M_i = \alpha_i + m_i$$

De cette façon, on génère n blocs de métadonnées. La méthode de chiffrement donné ci-dessus n'est qu'un exemple, le vérifieur peut prédéterminer une méthode plus forte pour le chiffrement.

- Concaténation de métadonnée et des données

Après l'étape de la génération, les blocs des métadonnées sont concaténés et ajoutés aux données tout en formant un ensemble de données $\tilde{F}$, cet ensemble de données sera envoyé au serveur pour le stocker.

Figure III-26: Données après la concaténation

- Vérification

Pour vérifier l'intégrité, le vérifieur demande un ensemble de position de bits dans un ensemble de blocs de données et leurs bits respectifs dans les métadonnées et les comparent pour faire la vérification.

Cette solution résout le problème du coût élevé de calcul et de la taille de données échangées lors de la vérification, mais le nombre de vérification reste très limité et la taille de données stockées sur le serveur dépend du nombre de bits choisis par blocs (par exemple pour un ensemble de données décomposé en n blocs, si on prend k bits par bloc, on doit concaténer $n*k$ bits au données), en plus cette solution peut déterminer si un fichier est intègre ou non mais ne peut pas déterminer la position de modification dans le cas de détection d'un problème.

3.2.4. Méthode 4 : Proofs of Retrievability for Large Files

Une troisième solution a été proposée par Juels, Ari, Kaliski Jr et Burton S [28]. Elle consiste à insérer des blocs « Sentinels » dans les données, elle se compose de 3 étapes résumées dans la figure suivante :

Figure III-27: Contrôle d'intégrité par insertion des Sentinels

- Fragmentation et cryptage des blocs

Le propriétaire fragmente les données en paquets et ajoute des codes correcteurs de l'erreur pour permettre par la suite la correction de blocs en cas de détection de problème d'intégrité, soit pour une raison de perte de données en transmission, soit par une attaque qui a modifié les données.

Cette méthode consiste à créer des blocs et les insérer dans des positions pseudo-aléatoires dans les données d'une façon qu'ils soient très difficile de les détecter. Pour cette raison le propriétaire des données doit prédéterminer la taille des blocs et le type d'algorithme de chiffrement permettant de chiffrer les données dans des tailles bien spécifiques, par exemple

AES-128, cet algorithme de chiffrement sera par suite appliqué séparément sur chaque bloc de données.

- Création, insertion et permutation des Sentinels avec les données

Les Sentinels sont des blocs générés pseudo-aléatoirement par une fonction définie par $f:\{0,1\}^j \times \{0,1\}^* \rightarrow \{0,1\}$, en utilisant cette fonction on génère un nombre s de Sentinels tel que $\{a_w\}_{w=1}^{s}\ et\ a_w = f(k,w)$, on ajoute ces Sentinels au données.

Après l'ajout des Sentinels à l'ensemble des données, une phase de permutation est nécessaire pour assurer que les Sentinels sont insérées dans des positions pseudo-aléatoires, ce qui impose l'utilisation d'une permutation pseudo-aléatoire g définie telle que $g:\{0,1\}^j \times \{1,\dots,b'+s\} \rightarrow \{1,\dots,b'+s\}$ comme illustré dans la figure III-28. De cette façon même en cas d'attaques, l'attaquant ne peut pas déterminer quel bloc appartient à la donnée originale et quel bloc appartient aux Sentinels.

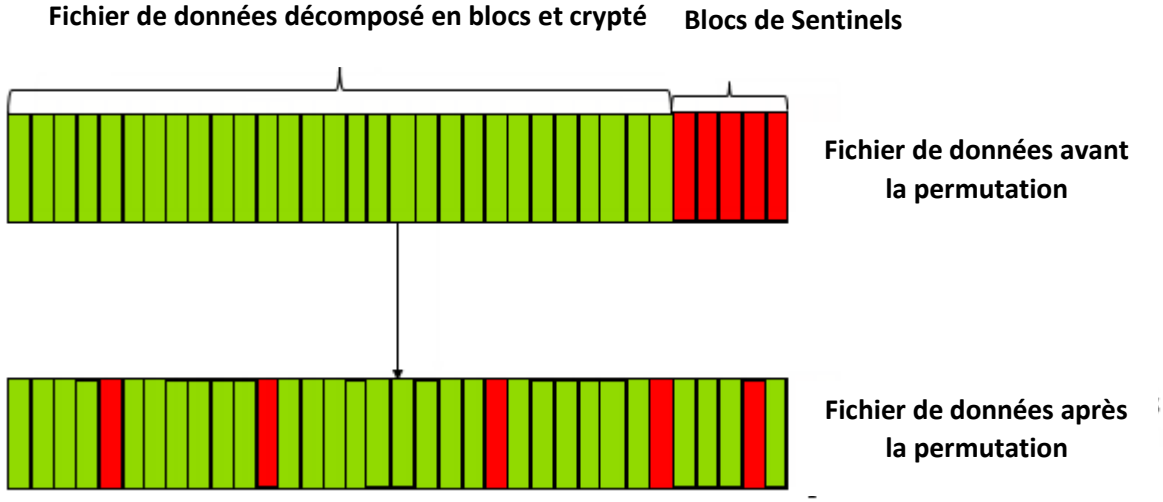


Figure III-28: La permutation des Sentinels

- Vérification

Dans la phase de vérification, le vérifieur utilise une fonction permettant - en prenant comme paramètre une variable σ - de donner la position du $\sigma^{ème}$ bloc de Sentinel, cette fonction initialise σ à 1 et répète ce processus p fois, puis elle envoie les valeurs trouvées au serveur qui va chercher par la suite les positions demandées et les retourner à l'utilisateur qui va tester si les blocs des Sentinels sont correctes.

Cette méthode permet de détecter et même de corriger des erreurs (avec un taux de correction relatif au code correcteur choisi), mais elle est utilisable seulement en cas de stockage passif des données car la modification et la lecture des données nécessitent la re-permutation des blocs et le déchiffrement qui est très coûteux. En plus, la vérification des données nécessite

le téléchargement d'un nombre *k* de blocs de taille *n* ce qui impose un téléchargement de taille *n*k* qui croît avec la taille des données et le nombre de Sentinels nécessaire pour la vérification.

3.2.5. Méthode 5 : Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage

Thomas Schwarz, S.J. Ethan et L. Miller [29] ont proposé de vérifier l'intégrité en se basant sur la signature algébrique, cette approche permet à un utilisateur tierce de vérifier l'intégrité sans avoir l'accès aux données, ce qui garde le principe de confidentialité.

Généralement une signature algébrique est définie telle que :

$$sig_{\alpha}(x_0, x_1, \dots, x_{N-1}) = \sum_{v=0}^{N-1} x_v . \alpha^v$$

Avec α un paramètre fourni par le propriétaire ou le vérifieur, cette signature peut être écrite sous forme d'une concaténation de plusieurs signatures comme suit :

$$sig_{(n,\alpha)} = (sig_{\alpha^0}, sig_{\alpha^1}, sig_{\alpha^2}, \dots, sig_{\alpha^{n-1}})$$

De cette façon, pour une α donnée on peut détecter les changements dans *n* symboles.

On adopte un code correcteur d'erreur qui va générer k vecteurs de parité $P_1, \dots, P_k$ à partir de *m* sous-ensemble de données $D_1, D_2, \dots, D_m$ tel que $P_i = \wp_i(D_1, D_2, \dots, D_m)$, on trouve que $sig_{\alpha}(\wp_i(D_1, D_2, \dots, D_m)) = \wp_i(sig_{\alpha}(D_1, D_2, \dots, D_m))$ (la preuve se trouve dans [30]). Cette propriété est implémentée dans cette méthode de telle façon à ce que lors de la vérification le vérifieur n'a qu'à demander la signature d'un ensemble de sous-ensembles de données et leur parité respective et vérifier si la propriété est toujours respectée.

- Le propriétaire des données calcule les parités des données et distribue les données avec leurs parités sur les serveurs distribués (figure III-29) :

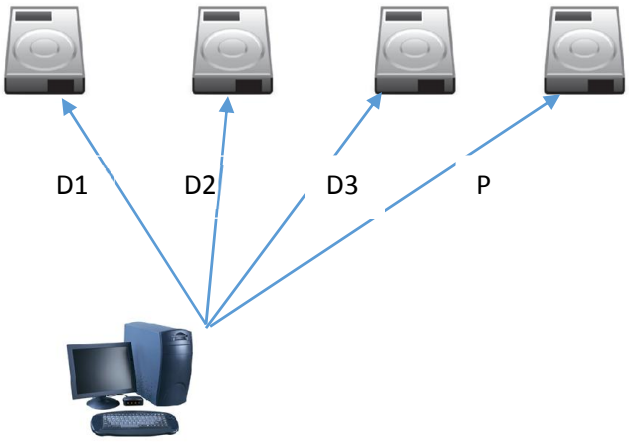


Figure III-29: Distribution des données sur les serveurs distribués

- Lors de la vérification, le client ou le vérifieur demande aux serveurs de calculer les signatures des données et la signature de la parité des données :

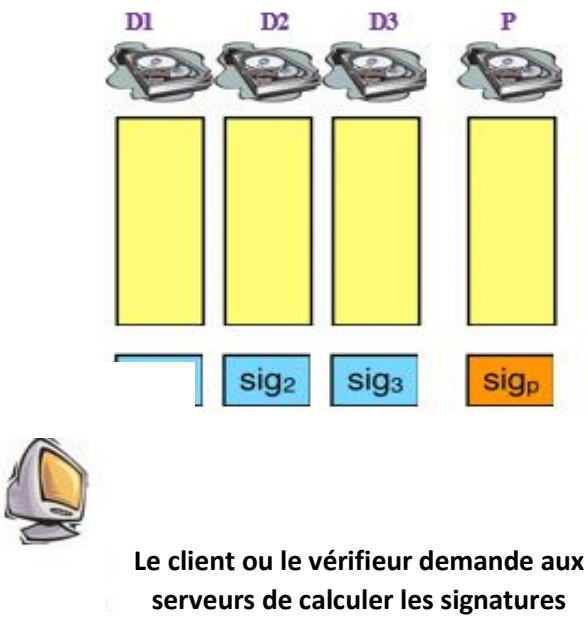


Figure III-30: Demande des signatures du serveur

- Le client/vérifieur reçoit les signatures et compare la somme des signatures des données avec la signature de la parité :

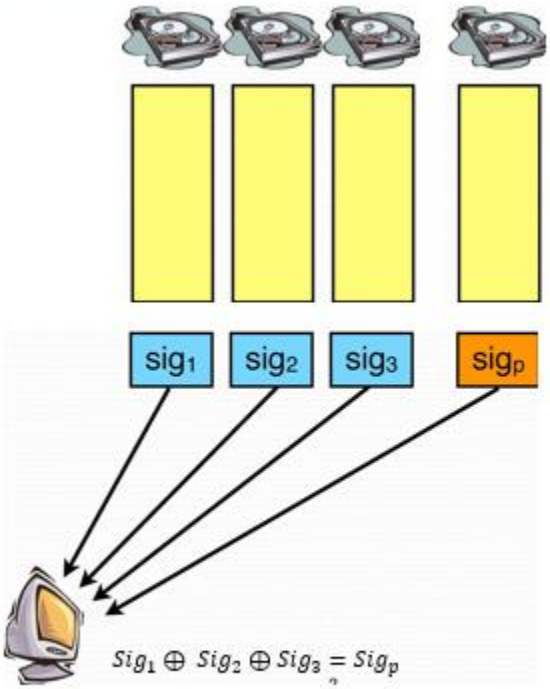


Figure III-31 : Réception des signatures et vérification de l'intégrité des données

3.2.6. Méthode 6 : Ensuring Data Storage Security in Cloud Computing

L'approche proposée par Cong Wang, Q.W., and Kui Ren et Wenjing Lou[31] se décompose en quatre parties principales :

- La distribution et la réparation des fichiers

  Pour permettre la réparation des fichiers, les données sont réparties sur *n* serveurs avec *n=(m + k)* et *k* étant les vecteurs de redondance de parité construits à partir de *m* vecteurs de données d'une façon qu'on peut reconstruire les vecteurs d'information *m* originaux à partir de n'importe quel *m* du *m+ k* vecteurs de parité dans les différents serveurs, ce qui garantit la possibilité de récupérer les données originales tant que le nombre de serveurs hors service est inférieur ou égale à ***k.***

- Pré-calcul des jetons pour les défis :

  Avant la distribution des données on pré-calcule un nombre de jetons pour chaque vecteur de données$G^{(j)}$ $(j \in \{1, \dots, n\}$, chaque jeton couvre un nombre aléatoire de sous-blocs d'informations, quand l'utilisateur veut vérifier les données, il envoie un ensemble d'indices générés aléatoirement aux serveurs qui vont calculer les signatures puis il vérifie si les signatures sont correctes ou non.

Algorithm 1 Token Pre-computation

procedure
  Choose parameters l, n and function f, ϕ;
  Choose the number t of tokens;
  Choose the number r of indices per verification;
  Generate master key K_{prp} and challenge k_{chal};
  for vector $G^{(j)}$, $j \leftarrow 1, n$ **do**
    for round $i \leftarrow 1, t$ **do**
      Derive $\alpha_i = f_{k_{chal}}(i)$ and $k_{prp}^{(i)}$ from K_{PRP}.
      Compute $v_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)]$
    end for
  end for
  Store all the v_is locally.
end procedure

Figure III-32: L'algorithme de pré-calcule des jetons

- Détection et localisation des erreurs :

  Lors des vérifications des données, l'utilisateur peut détecter les erreurs dans les données stockées dans les serveurs et il a même la possibilité de localiser le serveur dans lequel se situe l'erreur.

Algorithm 2 Correctness Verification and Error Localization

procedure CHALLENGE(i)
  Recompute $\alpha_i = f_{k_{chal}}(i)$ and $k_{prp}^{(i)}$ from K_{PRP};
  Send $\{\alpha_i, k_{prp}^{(i)}\}$ to all the cloud servers;
  Receive from servers:
  $\{R_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)] | 1 \le j \le n\}$
  for $(j \leftarrow m+1, n)$ **do**
    $R^{(j)} \leftarrow R^{(j)} - \sum_{q=1}^{r} f_{k_j}(s_{I_q,j}) \cdot \alpha_i^q, I_q = \phi_{k_{prp}^{(i)}}(q)$
  end for
  if $((R_i^{(1)}, \ldots, R_i^{(m)}) \cdot \mathbf{P} == (R_i^{(m+1)}, \ldots, R_i^{(n)}))$ **then**
    Accept and ready for the next challenge.
  else
    for $(j \leftarrow 1, n)$ **do**
      if $(R_i^{(j)} ! = v_i^{(j)})$ **then**
        return server j is misbehaving.
      end if
    end for
  end if
end procedure

Figure III-33: L'algorithme de vérification et de localisation des erreurs

- Récupération de fichiers et de correction d'erreur

Comme mentionné précédemment, la façon dans laquelle on a distribué les fichiers permet de récupérer les vecteurs des informations des serveurs et reconstruire l'information, ce qui fait, lors de la détection des erreurs on récupère les vecteurs correspondants et on les corrige à partir du reste des vecteurs et on les renvoi aux serveurs correspondants.

Algorithm 3 Error Recovery

procedure
  % Assume the block corruptions have been detected among
  % the specified r rows;
  % Assume $s \le k$ servers have been identified misbehaving
  Download r rows of blocks from servers;
  Treat s servers as erasures and recover the blocks.
  Resend the recovered blocks to corresponding servers.
end procedure

Figure III-34: L'algorithme de correction des erreurs

Cette solution permet de garantir l'intégrité et la confidentialité des données dans le Cloud, tant que les vecteurs des données erronées ne dépassent pas un nombre défini initialement *k* on peut récupérer, corriger et même détecter les serveurs malveillants ou contenant des problèmes affectant les données qu'ils contiennent.

Le tableau ci-dessous illustre une comparaison entre les différentes méthodes citées :

Tableau III-9:Comparaison entre les différentes méthodes citées dans cette section

Méthode	Utilisation des ressources	Nombre de vérifications	Permet la récupération des données endommagées	Utilisable sur les données dynamiques	Respecte le principe de confidentialité
Méthode de base 1	Elevée	N'est pas limité ; l'utilisateur récupère les données pour chaque vérification	Non	Non	Oui, la vérification est faite par l'utilisateur lui-même
Méthode de base 2	Elevée	N'est pas limité ; le serveur d'audit récupère les données pour chaque vérification	Non	Non	Non, les blocs sont envoyés au serveur d'audit tiers, ce qui lui expose les données
Data Integrity Proofs in Cloud Storage	Faible	Limité	Non	Non	Oui
Proofs of Retrievability for Large Files	Moyenne ; l'utilisation des ressources dépond de la taille des données et du nombre de blocs prédéfini pour chaque vérification	Limité ; le nombre de vérifications est prédéfini par l'utilisateur lors de la génération des blocs de « Sentinels »	Oui	Non	Oui
Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage	Faible ; la vérification est faite en comparant les signatures algébriques qui sont de taille faible et ne nécessitent	Très grand	Oui, mais très difficile	Non	Oui

	pas une utilisation importante de ressources de calculs				
Ensuring Data Storage Security in Cloud Computing	Faible ; l'utilisateur conserve seulement les jetons qui sont de taille faible de plus la vérification est faite par le serveur	Limité ; Le nombre des tests est proportionnel au nombre de signatures pré-calculées	Oui ; la méthode se base sur l'algorithme de Reed-Solomon qui permet de localiser et récupérer les données à partir des bits de parité pré-insérée	Oui; la mise-à-jour des données peut être faite en générant des matrices de mise-à-jour	Oui

Conclusion

Le Cloud Computing est un concept relativement nouveau qui présente un bon nombre d'avantages pour ses utilisateurs. Toutefois, il soulève également des problèmes de sécurité qui peuvent ralentir son utilisation. Comprendre les vulnérabilités existantes dans cette technologie va aider les organisations à faire la transition vers le Cloud.

Le Cloud Computing s'appuie sur de nombreuses technologies ce qui signifie qu'il hérite aussi leurs problèmes de sécurité. Nous avons présenté des problèmes de sécurité les plus fréquents pour les modèles de Cloud Computing: IaaS, PaaS et SaaS. On s'est également focalisé sur le stockage et la virtualisation qui sont les problèmes majeurs de la sécurité dans le Cloud Computing.

La sécurité de la virtualisation est l'une des préoccupations majeures pour les utilisateurs, cela est dû à son importance dans le Cloud. Les recherches dans ce cadre ont commencé depuis les années 60, plusieurs approches matérielles comme IOMMU et TPM et logicielles comme la vTPM ont été proposées pour remédier à la majorité des problèmes de sécurité.

La sécurité des données est la partie vitale dans la sécurité du Cloud, car dans la majorité des cas les attaques cible de violer soit l'intégrité soit la confidentialité des données. Pour remédier à ce problème plusieurs approches ont été proposées. Nous avons décrit dans ce mémoire quelque unes qui a permis de résoudre et faire face aux problèmes de l'intégrité et de la confidentialité, en prenant en compte les limitations du Cloud. Nous avons également établi une comparaison entre ces différentes méthodes en illustrant les points forts et faible de chaque approche.

Ce mémoire ouvre plusieurs perspectives. Dans un premier temps, nous comptons approfondir l'étude sur la confidentialité des données sur toutes les approches permettant l'indexation et la recherche des données chiffrées. En plus, les approche proposées dans la littérature ont permis de résoudre soit les problèmes de confidentialité soit les problèmes d'intégrité, ce qui ouvre la possibilité de penser à unifier la sécurité des données sous une seule approche permettant de garantir à la fois l'intégrité et la confidentialité des données tout en permettant la granularité du partage, la gestion des droits d'accès et la recherche indexée sur les données stockées.

Références

1. Mouhcine, B., *Sécurité dans les réseaux Ad Hoc*. 2012, Université Sidi Mohammed Ben Abdellah Faculté des Sciences Dhar Mahraz, Fès.
2. DOMAGE, E., *Security: 2010 Trends*. 2010: London.
3. Mell, P. and T. Grance, *The NIST definition of cloud computing (draft).* NIST special publication, 2011. **800**(145): p. 7.
4. informatique, S., *LE LIVRE BLANC DU CLOUD COMPUTING*, ed. S. informatique. 2010.
5. Ivan Studnia, E.A., Yves Deswarte, Mohamed Kaâniche, and Vincent Nicomette, *Survey of Security Problems in Cloud Computing Virtual Machines.* Computer and Electronics Security Applications, 2012.
6. P Clemente, L.d.B., J Rouzaud-Cornabas, *Security and Virtualization*. 2010, université d'orleans.
7. Sahoo, J., S. Mohapatra, and R. Lath, *Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues.* 2010: p. 222-226.
8. Clemente, P., L.E. de Bourges, and J. Rouzaud-Cornabas, *Security and Virtualization: a Survey.*
9. *Virtualization :a key to virtualization world*. 2012.
10. tim mather, S.K., Shahed Latif, *Cloud Security and Privacy*, ed. O.R. Media. 2009.
11. Inbarani, W.S., C.K.C. Paul, and W.A.J. Jeevakumar, *A Survey on Security Threats and Vulnerabilities In Cloud Computing.*
12. Subashini, S. and V. Kavitha, *A survey on security issues in service delivery models of cloud computing.* Journal of Network and Computer Applications, 2011. **34**(1): p. 1-11.
13. Lee, K., *Security Threats in Cloud Computing Environments.* International Journal of Security and Its Applications, 2012. **6**(4): p. 7.
14. Pankaj Arora, R.C.W., Er. Satinder Pal Ahuja, *Cloud Computing Security Issues in Infrastructure as a Service.* International Journal of Advanced Research in Computer Science and Software Engineering, 2012. **2**(1).
15. Wichers, D., *OWASP Top-10 2013.* 2010.
16. Tal Garfinkel, M.R., *When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments*
17. Chen, D. and H. Zhao. *Data security and privacy protection issues in cloud computing*. in *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*. 2012. IEEE.
18. Jayesh Dabhi , P.Z.Y.N., *Survey on Data Security Issues in Cloud Computing.* IJEDR, 2014. **2**(1).
19. xBrainLab, L.l., *Le Cloud Computing : Réelle révolution ou simple évolution ?* Wygwam.
20. Housley, R., et al., *RFC 2459: Internet X. 509 public key infrastructure certificate and CRL profile.* Network Working Group–Internet Engineering Task Force, 1999.
21. Perez, R., R. Sailer, and L. van Doorn. *vTPM: virtualizing the trusted platform module*. in *Proc. 15th Conf. on USENIX Security Symposium*. 2006.
22. Goyal, V., et al. *Attribute-based encryption for fine-grained access control of encrypted data*. in *Proceedings of the 13th ACM conference on Computer and communications security*. 2006. ACM.
23. Yu, S., et al. *Achieving secure, scalable, and fine-grained data access control in cloud computing*. in *INFOCOM, 2010 Proceedings IEEE*. 2010. Ieee.
24. Di Vimercati, S.D.C., et al. *Over-encryption: management of access control evolution on outsourced data*. in *Proceedings of the 33rd international conference on Very large data bases*. 2007. VLDB endowment.
25. Atallah, M.J., et al., *Dynamic and efficient key management for access hierarchies.* ACM Transactions on Information and System Security (TISSEC), 2009. **12**(3): p. 18.
26. Wang, W., et al. *Secure and efficient access to outsourced data*. in *Proceedings of the 2009 ACM workshop on Cloud computing security*. 2009. ACM.

27. Sravan Kumar, R. and A. Saxena. *Data integrity proofs in cloud storage*. in *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*. 2011. IEEE.
28. Juels, A. and B.S. Kaliski Jr. *PORs: Proofs of retrievability for large files*. in *Proceedings of the 14th ACM conference on Computer and communications security*. 2007. ACM.
29. Schwarz, T.S.J. and E.L. Miller, *Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage.* 2006: p. 12-12.
30. Litwin, W. and T. Schwarz. *Algebraic signatures for scalable distributed data structures*. in *Data Engineering, 2004. Proceedings. 20th International Conference on*. 2004. IEEE.
31. Cong Wang, Q.W., and Kui Ren,Wenjing Lou, *Ensuring Data Storage Security in Cloud Computing*. IEEE, 2009.