Master Cloud computing et calcul haute performance

Mémoire de Projet de Fin d'études

Etude et mise en place de chiffrement des données dans le cloud

Réalisée Par :
Nouhaila MOUSSAMMI
Aymane ANTAR

Encadré Par :
Pr. Ali Kartit [ENSAJ]
Membres de jury :
Pr Mostapha Zbakh [ENSIAS]
Président
Pr Ali Ouacha [FSR]
Examinateur

27 septembre 2021

Remerciements

Nous tenons à remercier toutes les personnes qui ont contribué au succés de notre projet fin d'études et qui nous ont aidées lors de la rédaction de ce rapport.

Avant d'entamer ce rapport nous profitons de l'occasion pour remercier tout d'abord notre professeur Monsieur Ali KARTIT qui n'a pas cessé de nous encourager pendant la durée du projet, ainsi pour sa générosité en matière de formation et d'encadrement. Nous le remercions également pour sa patience, sa disponibilité et surtout ses judicieux conseils qui ont contribué à alimenter notre reflexion.

Nous remercions aussi notre cher coordonnateur et professeur à l'université mohamed V monsieur Mostapha ZBAKH de nous avoir incités à travailler en mettant à notre disposition leurs expériences et leurs compétences.
Finalement, nous tenons à remercier nos parents pour leurs soutiens constants et leurs encouragements.

Résumé

Les entreprises et les particuliers ont de plus en plus besoin d'espace de stockage et de puissance de calcul. C'est la raison pour laquelle plusieurs technologies ont vu le jour comme le cloud computing. Le Cloud Computing, ou informatique en nuage, a émergé comme un nouveau paradigme pour offrir des ressources informatiques à la demande et pour externaliser des infrastructures logicielles et matérielles. Cette technologie fournit ses utilisateurs en espaces de stockage et en puissances de calcul en fonction de leurs besoins d'une manière flexible et personnalisée. De l'autre côté, la protection de la vie privée est un des enjeux majeurs de la société moderne dans laquelle Internet est omnipotent. Cependant, les données externalisées comme les photos, les emails, les rapports des entreprises sont souvent sensibles et confidentielles. Donc, il est primordial de protéger les données externalisées des attaques externes, ainsi que du serveur cloud lui-même. C'est pourquoi, il est très recommandé de chiffrer les données sensibles avant de les externaliser vers un serveur distant. L'objectif principal de ce travail est de mettre en œuvre cette approche de manière précise et fiable. La solution doit fournir un schéma de chiffrement permettant aux utilisateurs de conserver certaines fonctions et propriétés importantes, telles que la possibilité de calculs, la recherche, la modification et la préservation de l'ordre entre les données. Au premier lieu, Un modèle permettant de chiffrer les données avant de les stocker dans le cloud a été proposé, pour sécuriser les données stockées sur des serveurs externes. Ensuite, des systèmes de chiffrements homomorphes ont été pris en considération afin de permettre d'effectuer des opérations avec le contenu chiffré , sans avoir à le déchiffrer, dans le Cloud pour pouvoir bénéficier des forces de calculs du Cloud et éviter la saturation de la bande passante.

Mots clés - cloud computing, sécurité, chiffrements homomorphes, déchiffrement

Abstract

Increasingly, businesses and individuals need storage space and computing power. This is the reason why several technologies have emerged such as cloud computing. Cloud Computing has emerged as a new paradigm for delivering computing resources on demand and for outsourcing software and hardware infrastructure. This technology provides its users with storage space and computing power according to their needs in a flexible and personalized way. On the other hand, the protection of privacy is one of the major challenges of modern society in which the Internet is omnipotent. However, outsourced data like photos, emails, company reports are often sensitive and confidential. So, it is of utmost importance to protect the outsourced data from external attacks, as well as from the cloud server itself. Therefore, it is highly recommended to encrypt sensitive data before externalizing it to a remote server. The main objective of this work is to implement this approach accurately and reliably. The solution must provide an encryption scheme that allows users to maintain some important functions and properties, such as computability, search, modification and preservation of order between data. First, a model for encrypting data before storing it in the cloud has been proposed, to secure data stored on external servers. Then, homomorphic encryption systems were taken into consideration in order to allow operations to be carried out with the encrypted content, without having to decrypt it, in the Cloud in order to benefit from the computational strengths of the Cloud and avoid band saturation.

Keywords—cloud computing, security, Homomorphic Encryption, decryption

Table des matières

Table des figures

Introduction générale

Pendant si longtemps, la sécurité a été la caractéristique la plus importante que les chercheurs essaient de fournir en appliquant différentes méthodes et en mettant en œuvre différentes techniques. Récemment, le cryptage a présenté la méthode la plus efficace à appliquer afin de garantir une certaine sécurité, confidentialité et protection aux différents systèmes existants. De plus, le chiffrement homomorphe a retenu l'attention en raison des innombrables avantages qu'il offre [28][16].

Le chiffrement homomorphe, appliqué aux clouds publics [1][6], présente le concept d'un fournisseur de services qui peut effectuer différentes opérations sur des données chiffrées sans avoir besoin de les déchiffrer au préalable. Par conséquent, cette technique de cryptage est utilisée dans différents systèmes, à des fins différentes, elle fait preuve d'une grande adaptabilité et d'une grande aptitude tout en étant mise en œuvre de diverses manières.

Les cryptosystèmes homomorphes présentent un large champ d'application par rapport aux cryptosystèmes conventionnels, ce qui explique clairement leur énorme utilité. De plus, d'un point de vue théorique et pratique, il y a eu tellement d'applications dans de nombreux domaines de la sécurité pour le calcul en nuage en général, qui montrent des performances formidables lors de l'utilisation de cryptosystèmes homomorphes avec des techniques à profondeur limitée. Les schémas homomorphes ont la capacité de résoudre certains problèmes pratiques de cryptographie dans certains cas d'utilisation, tout en maintenant les données cryptées.

Le travail est organisé comme suit : Le 1er chapitre est une présentation des différentes notions en relation avec le cloud computing, à mentionner une brève explication de la notion de l'informatique dans le nuage, le modèle de déploiement et les caractéristiques essentielles.

Ensuite, le 2ème chapitre contient les aspects de sécurité dans le cloud, ensuite nous avons présenté les différentes solutions existantes. Finalement, l'objectif principal du 3ème chapitre est de présenter un tableau de comparaison entre les différents algorithmes utilisés reposant sur le cryptage homomorphe afin de conclure par la suite la technique la plus adéquate et la plus efficace en fonction du temps.

1 Généralités sur le Cloud Computing

1.1 Introduction

L'informatique en nuage, connu communément sous l'appellation «Cloud Computing» est un nouveau concept de la Technologie de l'Information qui a révolutionné ces dernières années le monde, et qui a attiré l'attention des chercheurs dans ce domaine, au lieu de déballer des ordinateurs et de les empiler dans une salle machine, le Cloud permet de télécharger virtuellement du matériel et de l'infrastructure associée à la demande.

Malgré qu'il n'a pas encore atteint le niveau de maturité attendu par ses utilisateurs, le marché du Cloud affiche des taux d'utilisation remarquables et intéresse de plus en plus les entreprises désireuses de booster leur système d'information et leur productivité. Son concept de base est d'externaliser des applications, des services ou des infrastructures informatiques vers des serveurs distants qui sont gérés par des fournisseurs de Cloud Computing. Il se réfère à l'utilisation des capacités de stockage et de calcul des serveurs répartis dans le monde entier, liés par le réseau internet. Certes, les utilisateurs ne sont plus propriétaires de leurs serveurs mais peuvent ainsi accéder de manière évolutive à de nombreux services en ligne sans avoir à gérer l'infrastructure sous-jacente, souvent complexe. Les applications et les données ne se trouvent plus en local, mais dans un nuage composé d'un grand nombre de serveurs distants interconnectés au moyen d'une grande

bande passante indispensable à la fluidité du système. Le Chapitre 1 introduit le concept du Cloud Computing, ces modèles de déploiement et ces services, et quelques avantages et incovenients.

1.2 Cloud Computing (CC)

1.2.1 Définiton

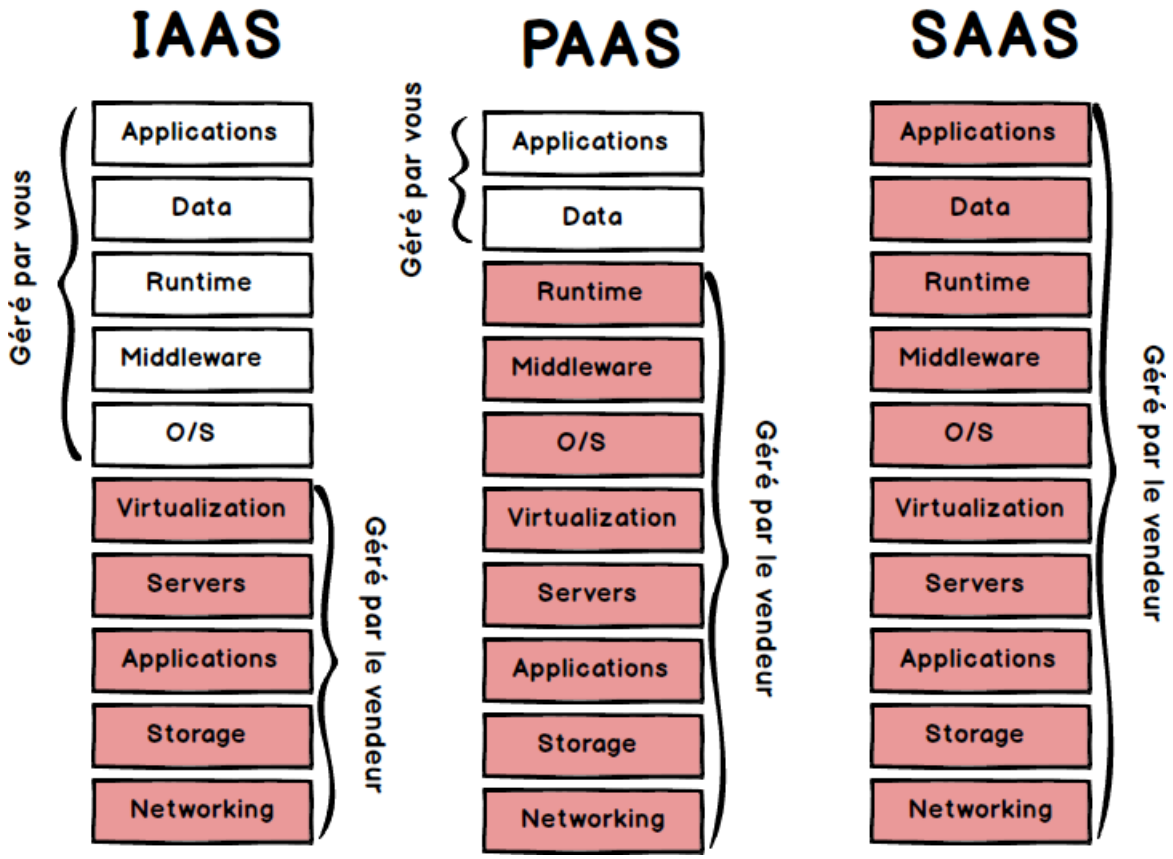
Le Cloud computing est un concept de déportation sur des serveurs distants des traitements informatiques traditionnellement localisés sur le poste utilisateur. L'idée est de déporter le traitement sur un serveur externalisé. Plus besoin d'installer le logiciel en local sur chaque poste, les fonctionnalités utiles pour l'entreprise se retrouvent toutes en ligne ou sur un serveur interne. Cette virtualisation des ressources permet à l'entreprise d'accéder à ses données sans avoir gérer une infrastructure informatique (. [30])
Cette technologie, permet aux utilisateurs de réduire leurs coûts d'exploitation des logiciels et du matériel directement en ligne, de gagner en performance et en extensibilité.

Le Cloud Computing permet de rendre une infrastructure matérielle et logicielle dynamique et flexible en exposant les capacités des Data-Centers comme étant un "réseau de services virtuels". Dans cette infrastructure, les utilisateurs peuvent accéder et déployer des applications à partir d'Internet suivant leurs demandes et la qualité de service exigée ([35]).

Le Cloud Computing offre à ses utilisateurs une grande capacité de calcul et de stockage. Les ressources offertes par le cloud sont disponibles sur le réseau et sont accessibles par n'importe quel appareil connecté à travers des mécanismes bien définis. Le Cloud peut servir plusieurs utilisateurs simultanément en affectant et en réattribuant les ressources. Des ressources de calcul, stockage, mémoire et d'autres ressources possibles sont regroupés dans des data centers et ne dépendent pas de l'emplacement de l'utilisateur. En d'autres termes, les utilisateurs du Cloud ne connaissent pas l'emplacement exact de la source des ressources fournies.

1.2.2 Modèles de services

IaaS PaaS et SaaS sont trois modèles de cloud computing populaires sur le marché. Selon les besoins, l'utilisateur peut choisir l'un des trois modèles.



Les tois modèles du Cloud Computing ([33]).

1.2.2.1 SaaS (Software as a Service)

Dans un SaaS, vous avez accès à des services d'application installés sur un serveur. Vous n'avez pas à vous soucier de l'installation, de la maintenance ou du codage de ce logiciel. Vous pouvez accéder au logiciel et l'utiliser avec votre navigateur uniquement. Vous n'avez pas à télécharger ni installer aucun type d'installation ou de système d'exploitation, le logiciel est simplement disponible pour que vous puissiez y accéder et le faire fonctionner. La maintenance, l'installation ou l'aide du logiciel sera fournie par le fournisseur de services SaaS et vous ne devrez payer que votre utilisation.([33])

1.2.2.2 PaaS (Plateform as a Service)

PaaS ou plate-forme en tant que service vous fournit des plates-formes informatiques qui incluent généralement le système d'exploitation, l'environnement d'exécution du langage de programmation, la base de données et le serveur Web. techniquement, c'est une couche au-dessus de l'IaaS, la deuxième chose que vous exigez après l'infrastructure est la plate-forme.

Les ordinateurs fournis dans une offre PaaS ont un système d'exploitation et des logiciels fixes. Vous pouvez exécuter votre logiciel en plus de cela.([33])

1.2.2.3 IaaS (Infrastructure as a Service)

IaaS fournit l'infrastructure telle que les machines virtuelles et d'autres ressources telles que la librairie d'images, disque de machines virtuelles, le stockage basé sur les blocs et les fichiers, les pare-feu, les adresses IP, les réseaux locaux virtuels, etc. L'infrastructure en tant que service ou IaaS est la couche de base du modèle en nuage.

De manière plus élaborée, IaaS est analogue à l'achat d'un meuble. On vous donne tous les matériaux, vous devez le construire vous-même. c'est-à-dire que vous recevez un ordinateur sans même un système d'exploitation et que vous pouvez installer le système d'exploitation et tous les logiciels qui s'y trouvent selon votre goût.([33])

1.2.3 Modèles de Déploiement

Nous distinguons deux formes principales de déploiements du Cloud Computing suivant le réseau dans lequel les services sont disponibles : Cloud public et Cloud privé, ainsi d'autre forme dérivée, à savoir Cloud hybride.

1.2.3.1 Cloud Public

Tout d'abord, les services du cloud peuvent être livrés par le biais de clouds publics auxquels l'accès est accordé publiquement pour tous types d'utilisateurs. Cela n'impose pas d'offrir la même qualité de service à tous les utilisateurs. Les fournisseurs peuvent toujours proposer différents plans cloud avec différents coûts en fonction de la quantité de ressources accessibles. Ces clouds ne peuvent être gérés que par les fournisseurs, ou par les institutions (entreprises, gouvernement, universités) qui les déploient dans les locaux des fournisseurs de services cloud. ([10])

Les fournisseurs du Cloud publique les plus connus sont Google, Microsoft et Amazon.

1.2.3.2 Cloud Privé

Contrairement aux clouds publics, qui accordent un accès ouvert au grand public, les clouds privés n'autorisent l'accès qu'à un ensemble fermé d'utilisateurs. Les services cloud privés sont exclusivement dédiés à un ensemble spécifique d'utilisateurs définis par le propriétaire du cloud. Ce type de déploiement est largement utilisé par les organisations professionnelles qui possèdent leurs propres clouds ou utilisent un cloud privé fourni par un fournisseur de services clouds. Les coûts de déploiement de ce modèle sont certes plus élevés mais il

est plus sécurisé ([10])

Eucalyptus, OpenNebula et OpenStack sont des exemples de solutions proposées pour la mise en place d'un Cloud privé.

1.2.3.3 Cloud Hybride

Toute combinaison de deux modes d'accès privé, public ou communautaire conduit à un mode d'accès en cloud hybride. Dans les clouds hybrides, deux clouds ayant deux ou plusieurs modes de déploiement sont opérationnellement liés. Ce type de déploiement est parfait pour pouvoir répartir ses moyens en fonctions des avantages recherchés.([10])

1.2.4 Les Avantages et les Incovénients

Avantages :

mises à jour et évolutivité : pas besoin de mettre à jour l'ensemble des postes pour ajouter de nouvelles fonctionnalités, il suffit de mettre à jour l'application réseau et tous les utilisateurs bénéficient des nouveautés et des corrections. Il en résulte une plus grande cohérence de la méthodologe de travail et des documents produits par l'ensemble des contributeurs de l'organistation.([30])
mise en commun des ressources : chaque utilisateur peut contribuer à l'enrichissement des données et des expériences de l'ensemble si des outils collaboratifs sont mis en place. Cet avantage facilite le kownledge management (gestion et transmission des connaissances) dans les entreprises([30]).
sécurité : si les documents ne sont plus présents en local (et que l'utilisateur ne sauvegarde pas ses identifiants de connexion sur son poste) on évite le problème de l'ordinateur perdu ou piraté et des documents confidentiels perdus dans la nature.([30])
puissance de calcul : le système déporté sur un réseau de serveurs offre une bien meilleure efficacité de calcul qu'un poste seul. Cette fonctionnalité est développée dans les domaines de la compression ou de l'application d'effets vidéo, plus largement dans le partage de vidéo (Youtube etc.) mais aussi dans le jeu en ligne (encore à l'état de test pour le grand public, le goulot d'étranglement étant la bande passante de l'utilisateur). Ce domaine est toutefois réellement intéressant dans le cadre d'application nécessitant une puissance de calcul importante pour une utilisation mobile.([30])
mobilité : l'utilisateur peut à tout moment et à partir de n'importe quel appareil se connecter à ses applications et son workflow. Il peut y accéder à partir de n'importe quel type d'appareil à condition que celui-ci soit doté d'une navigateur

Inconvénients :

sécurité : la plateforme cloud, si elle est externe (non installée sur le réseau interne ou avec une ouverture extérieure) doit être suffisamment sécurisée pour éviter le risque d'intrusion, de vol des données par piratage. L'autre risque est qu'un utilisateur oublie de se déconnecter sur un appareil accessible par des éléments externes à l'organisation. Il faut dans ce cas prévoir une déconnexion automatique en cas de non-activité du compte et bien segmenter les droits utilisateurs afin que ces derniers ne puissent accéder qu'aux données des projets dans lesquels ils sont impliqués. Plus généralement, une clause de confidentialité et la confiance dans son personnel sont primordiales pour que les données ne fuitent pas de manière volontaire.([30])
connexion : c'est l'autre goulot d'étranglement. Si l'utilisateur n'a pas de connexion internet, ou une connexion insuffisante, il ne pourra accéder à sa plateforme de travail. L'idée dans ce cas est de permettre le travail sur un application locale qui synchronise ensuite les données avec le serveur dès que l'utilisateur a à nouveau accès au réseau. Le problème de la sécurité des données en local se pose donc à nouveau([30])

1.3 Conclusion

Dans ce chapitre, nous avons abordé le concept du Cloud Computing, nous avons présenté la définition objective et générale de la notion de l'informatique dans le nuage, le modèle de déploiement et les caractéristiques essentielles. Cloud Computing est encore un paradigme en évolution, il intègre de nombreuses technologies existantes, c'est vrai qu'il est un modèle de prestation de services, où le logiciel, la plate-forme, l'infrastructure, les données, le matériel peut être livré directement en tant que service aux clients finaux. Une brève rétrospective de l'histoire de son évolution nous aide à clarifier les conditions, les opportunités et les défis existants dans son développement. Ces définitions, ces attributs et ces caractéristiques évoluent et changent au fil du temps.

2 Sécurité dans le cloud computing

2.1 Introduction

Le recours au Cloud est de plus en plus remarquable compte tenu de plusieurs facteurs, notamment ses architectures rentables, prenant en charge la transmission, le stockage et le calcul intensif de données. Il est devenu incontournable dans la mise en place et la fourniture des services informatiques. Parmi ces services on trouve le stockage externalisé des données. Storage as a service (STaaS) est un modèle permettant à un fournisseur de services de louer de l'espace dans son infrastructure de stockage à des individus ou à des entreprises. Au cours des dernières années, STaaS dans le Cloud a gagné en popularité chez les entreprises et les utilisateurs privés. Il permet à l'utilisateur final de profiter de la capacité de calcul maximale avec des configurations matérielles minimales requises. Toutefois, les questions de confidentialité des données, de sécurité, de fiabilité et d'interopérabilité doivent encore être résolues de manière adéquate. Mais le problème le plus important est la sécurité des données et la conformité aux réglementations, considérablement due à la perte de maîtrise et de gouvernance. Cet obstacle a limité la propagation de ce service. Les données stockées dans le Cloud peuvent être sensibles à l'entreprise et sont susceptibles d'être exploitées par le fournisseur lui-même ou d'autres personnes non autorisées. Actuellement, la plupart des utilisateurs du Cloud Storage protègent leurs données avec des contrats SLA. Ces Contrats sont basés, généralement, sur la confiance et la réputation du fournisseur. Cette faiblesse nous a motivé à réfléchir à des solutions qui permettent aux utilisateurs de sécuriser leurs données pour éviter leurs utilisations malveillantes

2.2 Les problèmes de sécurité dans le cloud computing

2.2.1 Principaux problèmes de sécurité

Le tableau 1, extrait de l'étude faite dans [5], met en évidence les principaux problèmes devant correctement être traités dans le cloud computing. Comme la colonne Total le montre, il en ressort que sur sept critères, six d'entre eux concernent plus de 70% des petites et moyennes entreprises (PME) interrogées dont la confidentialité des données est en tête des préoccupations. L'European Network and Information Security Agency (ENISA) a identifié trente-cinq risques pour le cloud computing listés dans [6]. Ces risques sont regroupés en quatre sections :

1) Risques organisationnels et politiques : tous les risques liés à la gestion du cloud, à son interopérabilité, aux respects de conformité, etc.

2) Risques techniques : tous les risques liés à l'utilisation d'un cloud : interception des données en transit, attaques, échec de l'isolation des ressources, etc.

3) Risques légaux : tous les risques liés au plan légal comme le changement de juridiction des données, etc.

4) Risques non spécifiques au cloud : ensemble des risques courants dans les systèmes informatiques comme la gestion du réseau (congestion, utilisation non optimale), la perte ou compromission de logs, etc.

De ces risques, l'ENISA en a extrait les huit plus importants dont cinq concernent directement ou indirectement la confidentialité des données : l'échec de l'isolation, la gestion des interfaces compromises, la protection des données, la suppression non sécurisée ou incomplète des données et les attaques de l'intérieur. Similairement à l'ENISA, le CSA a identifié treize domaines d'intérêts en matière de sécurité des clouds répartis en trois sections [18] :

1) Architecture du cloud : traite de l'architecture du cloud computing et de sa sécurisation ;

2) Administration du cloud : traite des domaines législatifs et légaux, des audits, du management et de l'interopérabilité/migration des clouds ;

3) Exploitation du cloud : traite principalement des problématiques de sécurité comme la gestion des incidents, des accès, du chiffrement, etc.

2.2.2 Quelques exemples d'attaques

Comme nous l'avons déjà mentionné, le cloud computing repose sur un ensemble de technologies préexistantes le rendant donc sensible aux attaques ciblant chacune de ces technologies. Par exemple, concernant le web 2.0, il existe un nombre important d'attaques comme le montre [9] qui réalise un état de l'art sur ces diverses techniques.
Ces attaques concernent principalement les couches PaaS et SaaS. Comme autres types d'attaques, nous pouvons citer par exemple :

«Joanna Rutkowska's Red and Blue Pill exploits» : cette attaque permet d'installer un backdoor sur un système cible par le biais de la virtualisation (IaaS) ;

«Kortchinksy's CloudBurst presentations» : cette attaque permet d'exécuter du code sur le système hôte à partir d'un système invité de ce host (IaaS) ;

«Distributed denial-of-service (DDoS)» , : cette attaque a pour but de rendre des services indisponibles (PaaS et SaaS).

En ce qui concerne les attaques des hyperviseurs serveur, bien qu'à l'heure actuelle elles restent marginales, il est fort probable qu'elles se généralisent d'ici quelques années. Ces attaques sont d'autant plus dangereuses qu'elles s'attaquent à la couche la plus basse du cloud computing et donnent la possibilité de mettre hors de service un cloud tout entier. De plus, la facilité d'utilisation d'un cloud permet de renforcer certaines attaques préexistantes comme :

«Zeus botnet» : Réussite de l'hébergement et de l'exécution du module central (command and control) du botnet Zeus spécialisé dans le vol de données sensibles (bancaires, mots de passe) sur la plateforme de cloud d'Amazon (IaaS et PaaS) ;

«Spam Attack» : Attaque issue de l'utilisation du cloud d'Amazon qui par le biais de spams à très grande échelle a permis d'installer des chevaux de Troie sur les machines cibles des spams. Cette attaque a pu être réalisée grâce à la facilité que procure le cloud d'obtenir des ressources informatiques (IaaS et PaaS).
Enfin, l'apparition du cloud computing a également fait naître de nouvelles attaques comme l'utilisation d'un déni de service distribué dont le but n'est plus de bloquer un service mais d'augmenter artificiellement les ressources allouées

pour le propriétaire du service par le cloud .

Ceci est réalisé dans le but d'augmenter considérablement les coûts d'utilisation du cloud pour le propriétaire du service afin de le mener à une faillite financière. Cette attaque peut être également réalisée sur un cloud entier afin de toucher tous les utilisateurs du cloud.

Ainsi, le cloud computing doit pouvoir se protéger contre de nombreuses attaques : celles issues de chacune des technologies préexistantes qui ont permis son émergence, mais également de celles qui ont évolué et de celles qui sont apparues en même temps que le cloud computing. De plus, comme il l'est souligné dans [17], il est nécessaire de définir de nouvelles méthodes d'évaluation des risques afin de prendre en compte l'aspect dynamique du cloud computing.

2.3 Fondamentale sur la Cryptographie

2.3.1 Définition

La cryptographie est la science qui utilise les mathématiques pour chiffrer et déchiffrer des données. La cryptographie vous permet de stocker des informations sensibles ou de les transmettre à travers des réseaux non sûrs (comme Internet) de telle sorte qu'elles ne puissent être lues par personne à l'exception du destinataire convenu.

Alors que la cryptographie est la science de la sécurisation des données, la cryptanalyse est la science de l'analyse et du cassage des communications sécurisées. La cryptanalyse classique mêle une intéressante combinaison de raisonnement analytique, d'application d'outils mathématiques, de découverte de redondances, de patience, de détermination, et de chance. Les cryptanalystes sont aussi appelés attaquants. La cryptologie embrasse à la fois la cryptographie et la cryptanalyse.

Depuis de nombreuses années, la cryptographie était le domaine exclusif des militaires, des services diplomatiques et gouvernementales secrètes, et a été utilisé pour fournir principalement des propriétés de sécurité, telles que la confidentialité des données, l'intégrité des données et l'authentification de l'origine des données [19]. Présentée comme l'art de codage de l'information dans les secrets, la cryptographie permet aux récepteurs de récupérer le contenu original des messages. Au cours de la deuxième partie du XXe siècle, le domaine de la cryptographie a élargi en raison de la prolifération des ordinateurs et des réseaux et l'apparition de nouveaux systèmes cryptographiques.

Un algorithme cryptographique, ou chiffre, est une fonction mathématique utilisée dans le processus de chiffrement et de déchiffrement. Un algorithme cryptographique fonctionne en combinaison avec une clé pour chiffrer le texte clair. Le même texte clair se chiffre en un texte chiffré différent si l'on utilise des clés différentes. La sécurité des données chiffrées est entièrement dépendante de deux choses : la force de l'algorithme cryptographique et le secret de la clé. Un algorithme cryptographique et toutes les clés possibles et tous les protocoles qui le font fonctionner constituent un cryptosystème.

En 1976, Diffie Hellman a effectué des changements radicaux dans la cryptographie, présentant le premier algorithme de cryptographie asymétrique [20]. En 1978, Rivest, Shamir et Adelman ont défini leur algorithme bien connu RSA [21]. Parallèlement, Koblitz et Miller ont proposé un nouveau schéma cryptographique basé sur des structures de courbes elliptiques. Récemment, la cryptographie quantique apparaît comme la cryptographie de l'avenir, car elle ne repose pas sur la théorie de l'algèbre et les groupes abstraits, mais sur les théories optiques et de lumière, où chaque bit est représenté par la polarisation d'un photon.

2.4 Crypto système

2.4.1 Définition

Un cryptosystème est un ensemble d'algorithmes permettant de chiffrer et déchiffrer des messages. Éventuellement, un cryptosystème pourra être accompagné d'un algorithme permettant la génération des clés de chiffrement et déchiffrement.

Cryptographie symétrique : Les algorithmes à clés symétriques sont une classe d'algorithmes de cryptographie qui utilisent les mêmes clés cryptographiques pour le chiffrement du texte en clair et le déchiffrement du texte chiffré. Les clés, dans la pratique, représentent un secret partagé entre deux ou plusieurs parties qui peuvent l'utiliser pour maintenir un lien d'information privée. Cette exigence que les deux parties doivent partager la clé secrète est l'un des principaux inconvénients de chiffrement à clé symétrique, par rapport au chiffrement à clé publique.

2.4.2 Cryptographie symétrique

2.4.2.1 Algorithme de chiffrement AES

L'AES (Advanced Encryption Standard) a été introduit par le NIST (National Institute of Standard and Technology) en 2001. Il s'agit d'un chiffrement par blocs symétriques et il a sans aucun doute été introduit pour surmonter la faiblesse du DES (Data Encryption Standard). Le chiffrement AES prend le bloc de texte brut de 128 bits accompagné d'une clé de 128 bits pour générer un bloc de texte chiffré de 128 bits. Et pendant le décryptage, il faut un bloc de texte chiffré de 128 bits avec une clé étendue utilisée dans le chiffrement dans l'ordre inverse pour récupérer le bloc de texte brut de 128 bits. AES étant un chiffrement par blocs symétriques, n'utilise pas la structure Feistel comme DES où le texte brut est divisé en deux moitiés. Et une moitié du bloc de texte brut aide à modifier l'autre moitié du bloc de texte brut, puis les deux moitiés sont permutées. AES fonctionne sur un bloc entier de texte brut sous la forme d'une seule matrice à chaque tour qui inclut la substitution et la permutation. AES a 10 tours -> clé de 128 bits 12 tours -> clé 192 bits 14 tours -> clé de 256 bits

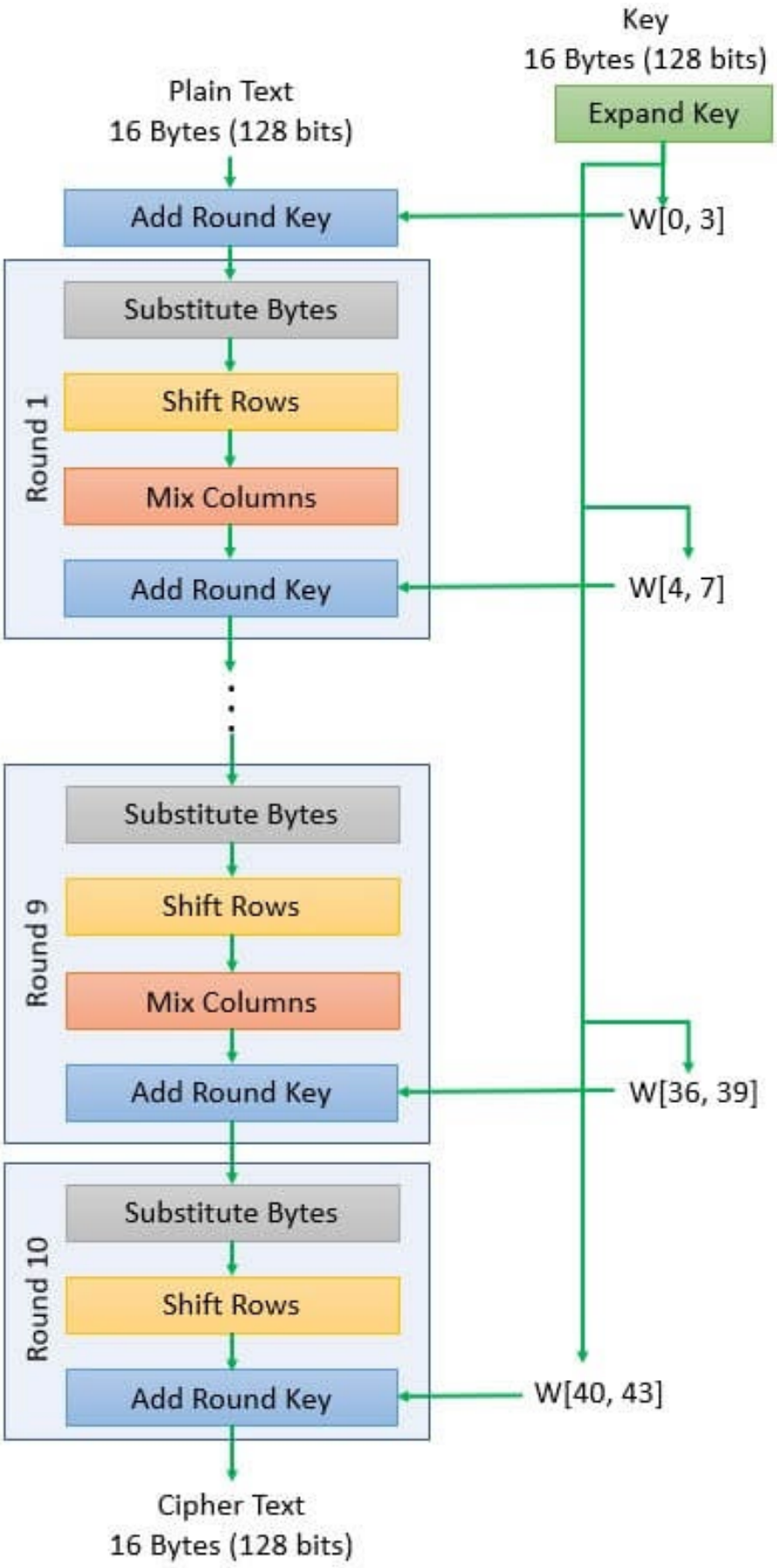


FIGURE 2.1 – Algorithme de chiffrement AES

Les avantages de l'AES sont nombreux : L'AES n'est pas sensible à toute attaque. Cependant, Brute Force attaque n'est pas une tâche facile, même pour un super-ordinateur. En effet, la taille de la clé de cryptage utilisée par l'algorithme AES est de l'ordre de 128, 192 ou 256 bits, ce qui conduit à des milliards de permutations et combinaisons. La rapidité et la faible exigence de la RAM étaient des critères du processus de sélection de l'AES [3]. AES fonctionne bien sur une grande variété de matériel, de 8 bits des cartes à puce à des ordinateurs de haute performance. AES est également beaucoup plus rapide que les algorithmes traditionnels.

2.4.3 Cryptographie asymétrique (à clé publique)

Dans le cas de la cryptographie asymétrique, chaque interlocuteur possède une paire de clés composée d'une clé publique (Public Key) et d'une clé privée (Private Key). Comme pour des clés physiques, ces clés sont liées comme dans un trousseau.
Dans ce cryptosystème, les deux clés sont étroitement liées à l'aide d'un algorithme mathématique : les données qui sont cryptées grâce à la clé publique peuvent uniquement être décryptées par la clé privée. Pour garantir la protection des données et le fonctionnement sans entrave de la cryptographie asymétrique, il est indispensable que la clé privée reste privée et ne soit connue de personne, pas même des autres interlocuteurs. En pratique, l'expéditeur de données à besoin de la clé publique du destinataire. La clé publique fonctionne à sens unique dans ce procédé : elle peut chiffrer les données, mais pas les déchiffrer, le déchiffrement n'est réalisable que par la clé privée du destinataire. La clé publique ne sert pas uniquement au chiffrement, elle permet également de vérifier une signature numérique et de vérifier les interlocuteurs.

La transmission de clés se fait lors du premier contact. Dans le même temps, la clé privée crée une signature numérique et peut ainsi être identifiée par les autres interlocuteurs. En d'autres termes, la cryptographie asymétrique permet que chaque participant puisse accéder à la clé publique, mais ne puisse décrypter les messages qu'avec la clé privée. Cela permet un échange de données hautement sécurisé. Un système de chiffrement asymétrique consiste alors en trois algorithmes :

1. KeyGen() : prend en entrée un paramètre de sécurité , retourne une paire de clés (pk ; sk) où pk désigne la clé publique utilisée pour chiffrer les messages et sk la clé privée utilisée pour déchiffrer.

2. Enc(m,pk) : prend en entrée le message m à chiffrer et la clé publique pk,

retourne un message chiffré c

3. Dec(c,sk) : prend en entrée un message chiffré c et la clé privée sk, retourne le message déchiffré m

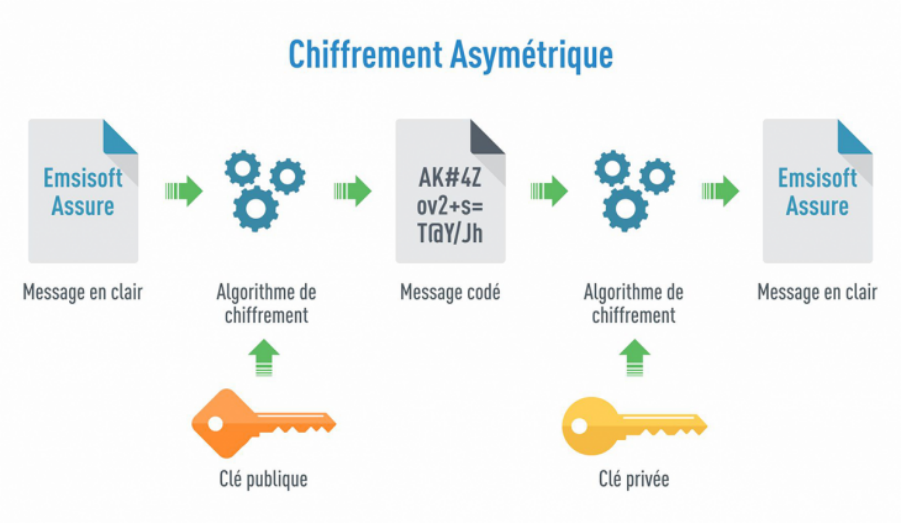


FIGURE 2.2 – Algorithme de chiffrement asymétrique ([3]).

Pour débuter la cryptographie asymétrique, le destinataire crée sa paire de clés. Il conserve la clé privée et permet à son ou ses interlocuteurs d'accéder à sa clé publique. Cela peut se faire par un simple transport, par une autorité de certification ou par des « Key Server » (c'est-à-dire des serveurs de clés) sur lesquels la clé peut être stockée. L'expéditeur utilise cette clé publique pour encoder son message et peut l'envoyer au destinataire sous forme de « texte chiffré ». À ce stade, le chiffrement du message n'est plus déchiffrable que par la clé privée du destinataire. C'est pour cette raison que le choix du canal de communication est en principe libre : si le message chiffré est intercepté, la personne qui récupère le message ne pourra pas accéder à son contenu.

C'est sur ce fonctionnement à sens unique que repose le système de cryptographie asymétrique. Les deux clés sont complètement indépendantes l'une de l'autre. Même lorsqu'il connaît la clé publique, un pirate ne peut pas en déduire d'informations sur la clé privée. Pour cela, les clés publiques fonctionnent selon des facteurs premiers clairement définis qui sont multipliés pour obtenir un résultat unique.

La clé privée travaille uniquement avec le résultat de cette opération. Il est presque impossible de tirer des conclusions sur les facteurs précédents à partir de cette valeur, car il existe d'innombrables possibilités quant à la manière dont cette valeur aurait pu être obtenue. À l'heure actuelle, il n'existe aucun procédé

mathématique ou algorithme qui simplifierait le calcul inversé. Un avantage important des chiffrements asymétriques par rapport aux chiffrements symétriques est qu'aucun canal secret n'est nécessaire pour pratiquer l'échange de la clé publique. Le récepteur doit simplement être sûr de l'authenticité de la clé publique. Les chiffrements symétriques demandent un canal secret pour envoyer la clé secrète – générée d'un côté du canal de communication – vers l'autre côté.

Les chiffrements asymétriques créent aussi moins de problèmes de gestion de clés que les chiffrements symétriques. 2n clés seulement sont nécessaires pour que n entités communiquent en toute sécurité entre elles. Dans un système basé sur des chiffrements symétriques, il faudrait n(n – 1)/2 clés secrètes. Dans une entreprise de 5 000 employés, par exemple, le déploiement généralisé d'une solution de sécurité basée sur le cryptage symétrique exigerait plus de 12 millions de clés. Avec une solution asymétrique, il n'en faudrait que 10 000.

Un inconvénient des chiffrements asymétriques par rapport aux chiffrements symétriques est qu'ils tendent à être environ « 1000 fois plus lents ». Autrement dit, il peut falloir plus de 1000 fois plus de temps de CPU pour traiter un cryptage ou décryptage asymétrique que symétrique.
Un autre inconvénient est que les chiffrements symétriques peuvent être percés par une attaque par « force brutale » : essai systématique de toutes les clés possibles jusqu'à trouver la bonne.

En raison de ces caractéristiques, les chiffrements asymétriques sont généralement utilisés pour l'authentification de données (par l'intermédiaire de signatures numériques), pour la distribution d'une clé de cryptage symétrique en masse (aussi appelée enveloppe numérique), pour des services de non-répudiation, et pour un accord de clé. Les chiffrements symétriques sont plutôt réservés au cryptage de masse.

2.4.3.1 Algorithme RSA

RSA est un système cryptographique, ou cryptosystème, pour le chiffrement à clé publique. Il est souvent utilisé pour la sécurisation des données confidentielles, en particulier lorsqu'elles sont transmises sur un réseau peu sûr comme Internet. RSA a été décrit pour la première fois en 1977 par Ron Rivest, Adi Shamir et Leonard Adleman du MIT (Massachusetts Institute of Technology). Le chiffrement à clé publique, également appelé chiffrement asymétrique, utilise deux clés différentes, mais mathématiquement liées, une publique et l'autre privée. La clé publique peut être partagée avec quiconque, tandis que la clé privée doit rester secrète. Dans le chiffrement RSA, tant la clé publique que la

clé privée peuvent servir à chiffrer un message. Dans ce cas, c'est la clé opposée à celle ayant servi au chiffrement qui est utilisée pour le déchiffrement. C'est notamment grâce à cette caractéristique que RSA est devenu l'algorithme asymétrique le plus répandu : il offre en effet une méthode permettant d'assurer la confidentialité, l'intégrité, l'authenticité et la non-répudiabilité des communications électroniques et du stockage de données.

De nombreux protocoles, tels que SSH, OpenPGP, S/MIME et SSL/TLS reposent sur RSA pour leurs fonctions de chiffrement et de signature numérique. Cet algorithme est également utilisé dans des logiciels : les navigateurs en sont un exemple flagrant, car établir une connexion sécurisée sur un réseau peu sûr comme Internet ou valider une signature numérique font partie de leurs attributions. La vérification de signature RSA constitue l'une des opérations les plus couramment réalisées en informatique. Si RSA offre autant de sécurité, c'est parce qu'il est très difficile de factoriser de grands entiers qui sont eux-mêmes le produit de deux grands nombres premiers. Multiplier ces deux nombres est facile, mais déterminer les nombres entiers d'origine à partir du total (la factorisation) est considéré comme une opération quasi impossible étant donné le temps qu'elle prendrait, même avec les ordinateurs super puissants actuels.

L'algorithme de génération des clés publique et privée est la partie la plus complexe du chiffrement RSA. Deux grands nombres premiers, p et q, sont générés à l'aide du test de primalité de Rabin-Miller. Un module de chiffrement n est calculé en multipliant p et q. Ce nombre est utilisé par les deux clés (publique et privée) et constitue le lien entre elles. Sa longueur, généralement exprimée en bits, est appelée la longueur de clé. La clé publique se compose du module n et d'un exposant public, e, en principe défini à 65537, car il s'agit d'un nombre premier pas trop élevé. Il n'est pas nécessaire que le nombre e soit un nombre premier secrètement choisi, car la clé publique est partagée avec tout le monde. La clé privée se compose du module n et de l'exposant privé d, qui est calculé à l'aide de l'algorithme d'Euclide étendu pour donner l'inverse modulaire par rapport à la valeur de l'indicatrice d'Euler en n.

La sécurité de RSA repose sur la difficulté que représente la factorisation de grands entiers. Cependant, à mesure que la puissance de traitement augmente et que des algorithmes de factorisation plus efficaces sont découverts, il devient possible de factoriser des nombres de plus en plus élevés. La puissance du chiffrement est directement liée à la taille de la clé. De ce fait, un doublement de la longueur de la clé renforce le chiffrement de façon exponentielle, au détriment toutefois des performances.

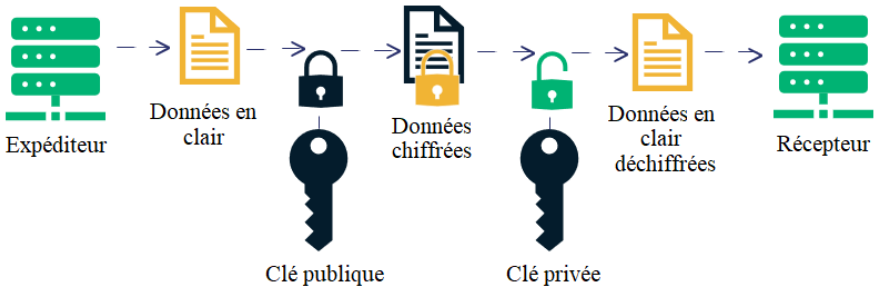


FIGURE 2.3 – Algorithme de chiffrement RSA.

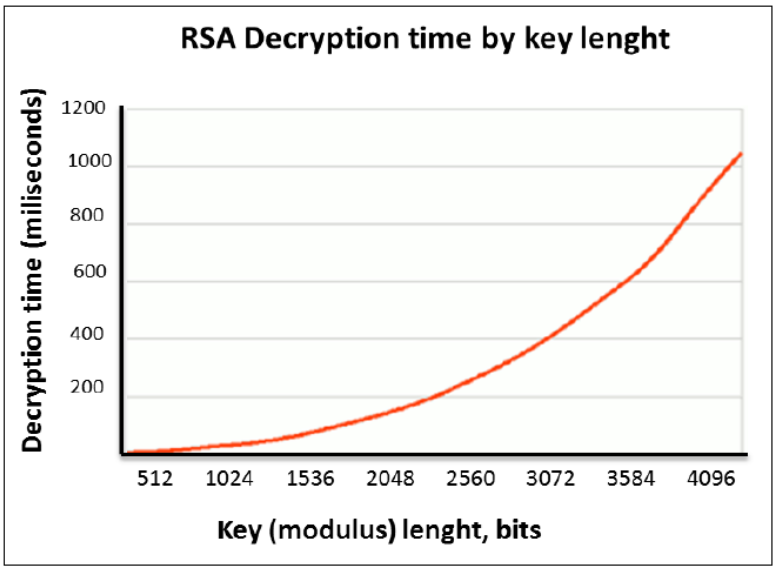


FIGURE 2.4 – Temps de déchiffrement par longueur de la clé RSA.

La sécurité de l'algorithme RSA repose sur deux conjectures. La première, considérer que pour casser le RSA et donc découvrir la clé privée, il faut factoriser le nombre n. La deuxième est de considérer que la factorisation est un problème difficile, c'est-à-dire qu'il n'existe pas d'algorithme rapide (de complexité polynomiale) pour résoudre cette question. Aucune de ces deux conjectures n'est prouvée. Il se peut donc que les deux soient fausses. Si c'est effectivement le cas, alors RSA n'est pas un algorithme de cryptographie sûr.

La génération de quadruplets (p, q, e, d) avec des nombres premiers p et q de quelques dizaines ou centaines de chiffres est une tâche facile et rapide, ce qui est essentiel pour une utilisation réelle du système cryptographique. On obtient sans peine des nombres premiers de cette taille par des algorithmes probabilistes de primalité. On peut ensuite choisir e au hasard et vérifier si le nombre obtenu est premier avec $(p - 1)(q - 1)$, ce qui est une opération facile et rapide en utilisant l'algorithme d'Euclide qui fournit dans le cas où e est premier avec $(p - 1)(q - 1)$ la clé privée d en même temps. Avec les techniques utilisées aujourd'hui pour programmer les systèmes RSA, on estime que le doublement

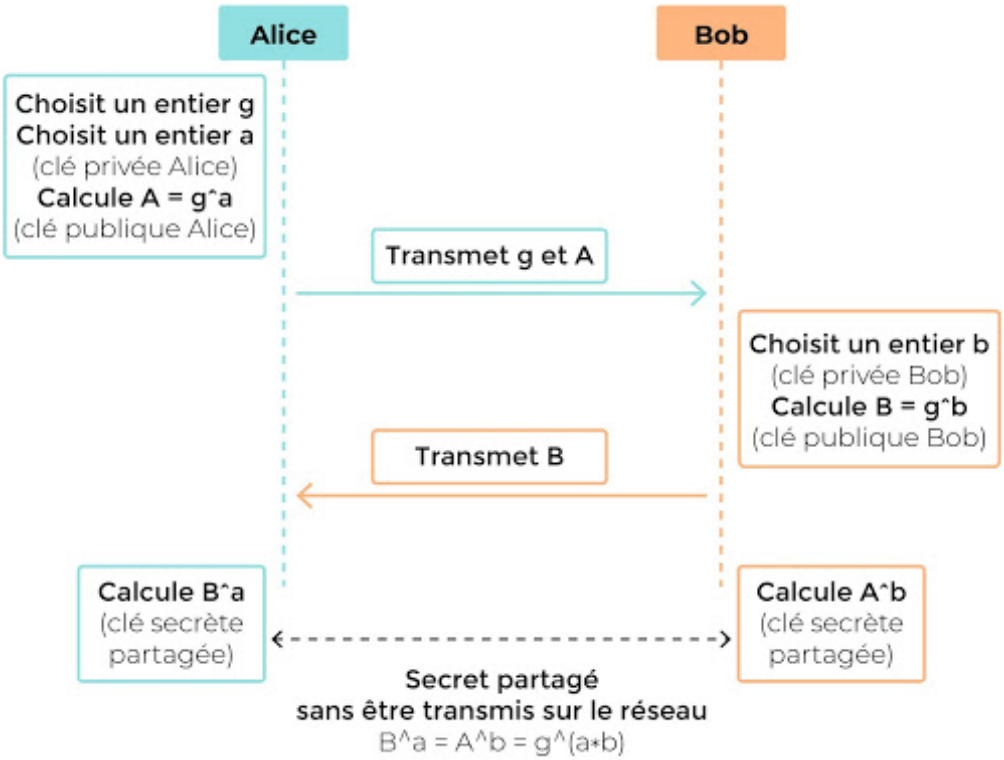
de la longueur des clés multiplie le temps de génération des clés par 16 et le temps de codage et décodage par 4.

L'usage du système RSA est donc dit polynomial : le temps de calcul se comporte comme un polynôme de la longueur de la clé ; en revanche, on fait l'hypothèse que casser le RSA nécessiterait des algorithmes bien plus longs que polynomiaux. Certains en déduisent que plus les machines seront puissantes, plus l'écart se creusera entre la puissance de calcul disponible mise en œuvre pour créer et utiliser des clés plus longues et la puissance requise pour casser le RSA. Autrement dit, plus le temps passe, plus le RSA devient robuste et sûr. La confiance dans la sécurité du système RSA ne repose pas sur une démonstration. Elle vient plutôt de l'échec répété depuis plus de 25 ans de toutes les tentatives pour casser ce système.

2.4.3.2 Sécurité RSA

2.4.3.3 Algorithme de Diffie-Hellman

L'échange de clés Diffie-Hellman est une méthode de cryptage spécifique développée par Whitfield Diffie et Martin Hellman et publiée en 1976, l'une des premières mises en œuvre dans le domaine de la cryptographie. La méthode d'échange de clés Diffie-Hellman permet à deux parties, qui ne se connaissent pas a priori, de partager une clé secrète sous un canal de communication non sécurisé. Une telle clé peut être employée pour chiffrer les messages ultérieurs en utilisant un schéma de chiffrement à clé symétrique.



L'échange de clés Diffie-Hellman consiste à échanger une clé secrète entre Alice et Bob sur un réseau non sécurisé de manière confidentielle. Il se déroule comme ceci :

1- Alice choisit un entier aléatoire g et un entier aléatoire de grande taille a qui est sa clé privée.

2- Elle calcule A = $exp(a) = g^a$ g à la puissance a . a est la clé privée d'Alice et A est la clé publique d'Alice.

3- Alice transmet à Bob la valeur de g et de A, sa clé publique.

4-De même, Bob choisit un entier aléatoire de grande taille b. Il calcule B = $exp(b) = g^b$. b est la clé privée de Bob et B est la clé publique de Bob.

5-Bob transmet à Alice B, sa clé publique.

6- À ce moment, Alice calcule $B^a = g(b * a)$ et Bob calcule $A^b = g^{(a*b)}$. Ces deux variables ont la même valeur, car $x^a * b = x^b * a$. Cette variable B^a est la clé secrète partagée par Alice et Bob.

Imaginons maintenant qu'une tierce personne, qu'on appellera Ève, ait écouté toutes les communications entre Alice et Bob. Ève connait g, A = g^a et B = g^b.

Cependant, Ève ne peut pas calculer a car la fonction log(A) = $log(g^a)$ = a est impossible à calculer. De même, elle ne peut pas calculer b. Ève ne peut donc pas calculer B^a ou A^b. Cette variable est donc confidentielle entre Alice et Bob, qui peuvent l'utiliser comme clé secrète partagée pour chiffrer leurs échanges de manière confidentielle avec le chiffrement symétrique.

2.4.3.4 PGP de Philip Zimmermann

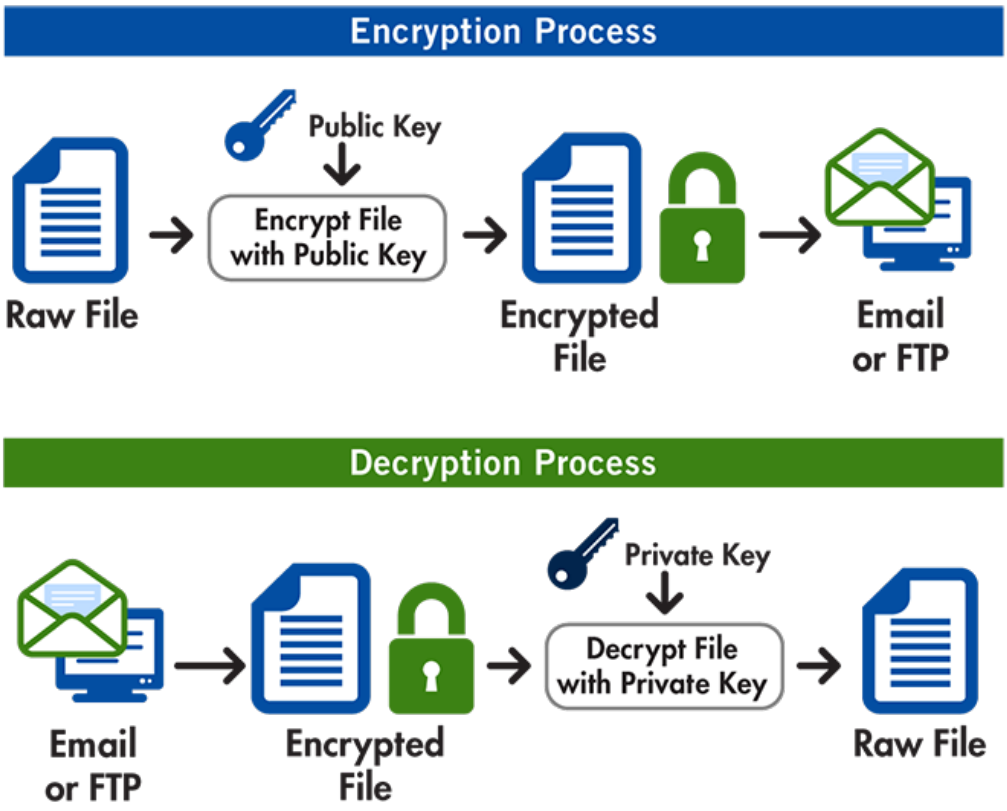


FIGURE 2.5 – Algorithme de chiffrement PGP.

PGP [Pretty Good Privacy] est un cryptosystème inventé par Philip Zimmermann, il combine à la fois les meilleures fonctionnalités de la cryptographie conventionnelle et de la cryptographie à clé publique. PGP est un cryptosystème hybride.

Quand un utilisateur chiffre du texte clair avec PGP, PGP compresse d'abord le texte clair. La compression de données économise du temps de transmission par modem et de l'espace disque et, ce qui est plus important, renforce la sécurité cryptographique. La plupart des techniques de cryptanalyse exploitent les redondances trouvées dans le texte clair pour craquer le texte chiffré. La compression réduit ces redondances dans le texte clair, ce qui augmente grandement la résistance à la cryptanalyse.
PGP crée ensuite une clé de session, qui est une clé secrète qui ne sert qu'une

fois. Cette clé est un nombre aléatoire généré à partir des mouvements aléatoires de la souris et des touches du clavier sur lesquelles on tape. Cette clé de session fonctionne avec un algorithme de chiffrement conventionnel très sûr et rapide qui chiffre le texte clair ; le résultat est le texte chiffré. Une fois que les données sont chiffrées, la clé de session est elle-même chiffrée avec la clé publique du destinataire. Cette clé de session chiffrée par la clé publique est transmise avec le texte chiffré au destinataire.

Le déchiffrement fonctionne de la manière inverse. La copie de PGP du destinataire utilise la clé privée de celui-ci pour retrouver la clé de session temporaire, que PGP utilise ensuite pour déchiffrer le texte chiffré de manière conventionnelle.

La combinaison des deux méthodes de chiffrement associe la commodité du chiffrement à clé publique avec la vitesse du chiffrement conventionnel. Le chiffrement conventionnel est environ 1000 fois plus rapide que le chiffrement à clé publique. Le chiffrement à clé publique fournit quant à lui une solution aux problèmes de distribution de la clé et de transmission des données. Utilisées toutes les deux, la performance et la distribution de la clé sont améliorées sans aucun sacrifice sur la sécurité.

Les clés sont stockées sous forme chiffrée. PGP stocke les clés dans deux fichiers sur votre disque dur ; l'un pour les clés publiques et l'autre pour les clés privées. Ces fichiers sont appelés des trousseaux de clés. Quand vous utiliserez PGP, vous ajouterez les clés publiques de vos correspondants à votre trousseau public. Vos clés privées sont stockées dans votre trousseau privé. Si vous perdez votre trousseau privé, vous serez incapable de déchiffrer tout message destiné à l'une des clés qui était dans ce trousseau.

2.5 Signatures numériques

Les signatures numériques sont comme des «empreintes digitales» électroniques. Sous la forme d'un message codé, la signature numérique associe de manière sécurisée un signataire à un document dans une transaction enregistrée. Les signatures numériques utilisent un format standard et accepté, appelé infrastructure à clé publique (PKI), pour fournir les plus hauts niveaux de sécurité et d'acceptation universelle. Il s'agit d'une implémentation de technologie de signature spécifique de la signature électronique (eSignature).

Une signature numérique est un schéma mathématique permettant de vérifier l'authenticité de messages ou de documents numériques. Une signature numé-

rique valide, où les conditions préalables sont satisfaites, donne au destinataire de très bonnes raisons de croire que le message a été créé par un expéditeur connu (authentification) et que le message n'a pas été altéré en transit (intégrité).

Les signatures numériques utilisent la cryptographie asymétrique. Dans de nombreux cas, ils fournissent une couche de validation et de sécurité aux messages envoyés via un canal non sécurisé : correctement mise en œuvre, une signature numérique donne au destinataire des raisons de croire que le message a été envoyé par l'expéditeur revendiqué. Les signatures numériques sont équivalentes aux signatures manuscrites traditionnelles à bien des égards, mais les signatures numériques correctement mises en œuvre sont plus difficiles à falsifier que le type manuscrit. Les schémas de signature numérique, au sens utilisé ici, sont basés sur la cryptographie et doivent être mis en œuvre correctement pour être efficaces. Les signatures numériques peuvent également fournir une non-répudiation, ce qui signifie que le signataire ne peut pas affirmer avec succès qu'il n'a pas signé un message, tout en affirmant que sa clé privée reste secrète. En outre, certains schémas de non-répudiation offrent un horodatage pour la signature numérique, de sorte que même si la clé privée est exposée, la signature est valide. Les messages signés numériquement peuvent être tout ce qui peut être représenté sous forme de chaîne de bits : les exemples incluent le courrier électronique, les contrats ou un message envoyé via un autre protocole cryptographique.

2.6 Fonctions de hachage

Une fonction de hachage cryptographique est une fonction de hachage qui, à une donnée de taille arbitraire, associe une image de taille fixe, et dont une propriété essentielle est qu'elle est pratiquement impossible à inverser, c'est-à-dire que si l'image d'une donnée par la fonction se calcule très efficacement, le calcul inverse d'une donnée d'entrée ayant pour image une certaine valeur se révèle impossible sur le plan pratique. Pour cette raison, on dit d'une telle fonction qu'elle est à sens unique.

En raison de leur ubiquité, ces fonctions à sens unique ont été appelées les chevaux de trait de la cryptographie moderne. La donnée d'entrée de ces fonctions est souvent appelée message ; la valeur de sortie est souvent appelée valeur de hachage, empreinte numérique, empreinte, ou encore haché (en anglais, message digest ou digest, hash).

Une fonction de hachage cryptographique idéale possède les quatre propriétés suivantes : La fonction est déterministe, c'est à dire qu'un même message

aura toujours la même valeur de hachage ; La valeur de hachage d'un message se calcule « facilement » ; Il est impossible, pour une valeur de hachage donnée, de construire un message ayant cette valeur (résistance à la préimage) ; Il est impossible de trouver un second message ayant la même valeur de hachage qu'un message donné (résistance à la seconde préimage) ; Il est impossible de trouver deux messages différents ayant la même valeur de hachage (résistance aux collisions) ; Modifier un tant soit peu un message, modifie considérablement la valeur de hachage.

Les fonctions de hachage cryptographiques sont une primitive de cryptographie symétrique, mais forment une brique de base largement utilisées dans la conception de schémas cryptographiques (aussi bien symétriques ou asymétrique), notamment dans les signatures numériques, les codes d'authentification de message et les autres formes d'authentification.

2.7 Certificats numériques

Les certificats numériques permettent l'identification unique d'une entité ; il s'agit de cartes d'identité électronique émises par des parties dignes de confiance. Les certificats numériques permettent à un utilisateur de vérifier l'identité de la personne à laquelle le certificat est délivré ainsi que l'identité de l'émetteur du certificat.

Les certificats numériques sont le vecteur utilisé par SSL pour la cryptographie de clé publique. La cryptographie de clé publique utilise deux clés cryptographiques différentes : une clé privée et une clé publique. La cryptographie de clé publique est également connue sous le nom de cryptographie asymétrique, car vous pouvez chiffrer les informations avec une clé et les déchiffrer avec la clé complémentaire d'une paire de clés publique-privée donnée.

Les paires de clés publique-privée sont simplement de longues chaînes de données qui jouent le rôle de clés dans le schéma de chiffrement d'un utilisateur. L'utilisateur conserve la clé privée dans un endroit sécurisé (par exemple, le disque dur d'un ordinateur) et fournit la clé publique à quiconque avec qui il souhaite communiquer. La clé privée permet de doter d'une signature numérique toutes les communications sécurisées émanant de l'utilisateur ; la clé publique est utilisée par le destinataire pour vérifier la signature de l'expéditeur.

La cryptographie de clé publique s'appuie sur la confiance ; en effet, le destinataire d'une clé publique doit être prêt à croire que la clé appartient réellement à l'expéditeur et non à un imposteur. Les certificats numériques offrent cette

confiance.

Le rôle d'un certificat numérique est double : il établit l'identité du propriétaire d'une part, et il rend la clé publique du propriétaire disponible d'autre part. Un certificat numérique est émis par une autorité digne de confiance et uniquement pour une durée déterminée. Une fois sa date d'expiration passée, le certificat numérique doit être remplacé.

2.8 Conclusion

Le Cloud computing est une technologie très prometteuse permettant à ses clients de réduire les coûts d'exploitation, d'administration etc. Tout en augmentant l'efficacité, toutefois, l'adoption de technologie reste faible, et cela revient aux problèmes de sécurité en particulier la sécurité des données échangées sur le réseau internet. Afin de résoudre ces problèmes et d'améliorer l'adoption et l'utilisation de cette technologie. Nous avons étudié les aspects de sécurité du cloud, ensuite nous avons présenté les différentes solutions existantes, par la suite nous allons présenter en détail une solution basée sur le chiffrement homomorphe.

3 Chiffrement homomorphe et préservant de l'ordre

3.1 Introduction

Chaque jour, les entreprises gèrent de nombreuses informations sensibles, notamment les informations personnelles et financières, qui doivent être chiffrées lors de leur stockage (données au repos) comme lors de leur transmission (données en transit). Les algorithmes de chiffrement modernes sont quasiment inviolables, pour le moins d'ici l'arrivée de l'informatique quantique, car une trop grande puissance de calcul est nécessaire pour les casser. En d'autres termes, cette opération n'est pas faisable en raison de l'argent et du temps requis. Chiffrer des données pose un problème, dans le sens où celles-ci devront être déchiffrées à plus ou moins long terme. En outre, le déchiffrage des données les rend vulnérables aux pirates. Vous pouvez stocker vos fichiers encodés cryptographiquement sur le cloud à l'aide d'une clé secrète, mais dès que vous avez besoin d'utiliser ces fichiers, par exemple modifier un fichier Word ou interroger une base de données financière, vous devez déverrouiller les données et les rendre vulnérables. Le chiffrement homomorphe, une avancée de la science cryptographique, pourrait bien faire évoluer les choses.

3.2 Énoncé du problème et objectifs

L'heure est à l'externalisation des données. C'est le cloud computing, qui permet de mutualiser les structures de conservation et de traitement des données. L'un des problèmes de ceci est la préservation de la confidentialité entre le client et le fournisseur. Certes, la cryptographie usuelle peut répondre en partie à ce problème, puisque le client peut décider de ne stocker que des données chiffrées.

Mais ceci n'est pas réaliste, comment faire par exemple, si vous voulez effectuer une recherche dans vos données ? Impossible, normalement, de le faire

directement sur les données chiffrées ! Sauf si votre système de chiffrement dispose de la propriété suivante : les algorithmes de traitement de données passent à travers la couche de chiffrement. Autrement dit, si le client veut effectuer certains calculs sur ces données, il suffit qu'il demande au fournisseur d'effectuer ces calculs sur les données chiffrées, le fournisseur transmet le résultat (qui est chiffré), le client le déchiffre et il obtient le résultat voulu (en clair). On appelle ces chiffrements des chiffrements homomorphes.

Le but du chiffrement homomorphe est d'autoriser les calculs sur les données chiffrées. Les données peuvent ainsi demeurer confidentielles lors de leur traitement afin de pouvoir effectuer des tâches utiles sur les données stockées dans des environnements non sécurisés. À l'époque des calculs distribués et des réseaux hétérogènes, il s'agit sans nul doute d'un avantage décisif.

Un cryptosystème homomorphe est similaire aux autres formes de chiffrement public, dans le sens où il emploie une clé publique pour chiffrer les données et autorise uniquement la personne disposant de la clé privée correspondante à accéder aux données déchiffrées. Ce qui la différencie toutefois des autres formes de chiffrement est son recours à un système algébrique pour permettre à une personne d'effectuer un ensemble de calculs (ou d'opérations) sur les données chiffrées.

3.3 travaux connexes

Dans cet rapport, nous avons discuté la sécurité des données dans l'état de traitement est l'un des principaux obstacles à l'adaptation du cloud computing. Nous avons également discuté trois méthodes de chiffrement homomorphe partiel et leurs applications dans le cloud computing. Les techniques de cryptage traditionnelles ne sont pas suffisantes pour sécuriser les données en cours de traitement. Il semble que les méthodes de cryptage homomorphes diminuent le problème de la sécurité des données dans le cloud computing. Mais actuellement, les méthodes homomorphes tant complètes que partielles ne sont pas réalisables et pas si faciles à mettre en œuvre pour le cloud computing. Les auteurs de [10] ont montré que la méthode entièrement homomorphe prenait beaucoup de temps pour opérer sur des données cryptées. D'un autre côté, les méthodes homomorphes partielles ne sont pas non plus efficaces car elles sont capables d'effectuer soit une addition soit une multiplication [22].

Les performances de tout schéma de chiffrement sont évaluées selon plusieurs critères. Les plans SHE existants ne sont pas évalués efficacement par rapport aux critères d'opportunité, de simplicité et de sécurité. Par conséquent, il est

encore nécessaire d'améliorer l'évaluation de ces critères. Même si certains des problèmes de sécurité [29][5] dans le schéma d'origine sont par la suite résolus, il existe encore des problèmes de sécurité ouverts concernant les schémas SHE. Il y a également une perte de confidentialité pour les utilisateurs, et ces appareils sont connus pour être envahissants. Ainsi, certains problèmes d'expansion des données et d'augmentation du bruit ne sont toujours pas résolus. Il est donc nécessaire de proposer un schéma dans le domaine homomorphe offrant une réduction de bruit significative et de résoudre le problème d'expansion des données.

Plus tard, les chercheurs commencent à faire des tests en appliquant un type d'action mathématique, soit l'addition, soit la multiplication, par exemple le schéma RSA. Jusqu'en 2008, avec les travaux avancés sur le chiffrement homomorphe, nous sommes devenus capables d'appliquer différents types d'opérations algébriques, connus d'avance, ce qui a conduit à l'apparition du chiffrement quelque peu homomorphe. Ce type de schéma est basé sur l'addition et la multiplication homomorphes. Mais il prend en charge un nombre limité d'opérations pour réduire le bruit et ne pas perdre le texte en clair initial après le déchiffrement. Ainsi, l'application d'un certain nombre d'opérations mathématiques rend le système pratique et peut être mis en œuvre pour des tests de performances. Donc, si nous voulons tester une technique spéciale, nous pouvons utiliser un schéma basé sur SHE pour nous assurer des mérites que cette technique pourrait apporter, puis l'appliquer à des schémas plus complexes.

En 2009, Gentry a présenté la première étape du cryptage entièrement homomorphe [2]. Ce dernier est considéré comme le schéma le plus sécurisé et le plus structuré puisqu'il rassemble à la fois les mérites de différentes opérations mathématiques en cloud computing. Par ailleurs, FHE était initialement basé sur un schéma SHE associé à la technique de bootstrap pour réduire le bruit. Cette technique empêche l'expansion de la taille du texte chiffré, mais elle est encore peu pratique. Cependant, jusqu'à présent, il n'existe pas de schéma FHE pratique d'un point de vue performance. Lors de la mise en œuvre de tels schémas, on remarque une extension de la taille des données et il ne prend pas en charge plusieurs utilisateurs. En plus de cela, nous devons utiliser une grande variété de paramètres selon l'application qui

3.4 Solutions de sécurité de données dans le Cloud

Plusieurs techniques et solutions ont été proposées pour sécuriser les données dans le Cloud. Ces solutions peuvent être classées selon plusieurs critères, comme illustré dans la figure suivante, où la taxonomie des techniques de sécu-

rité repose sur la stratégie de sécurité et le principe de chiffrement inspiré

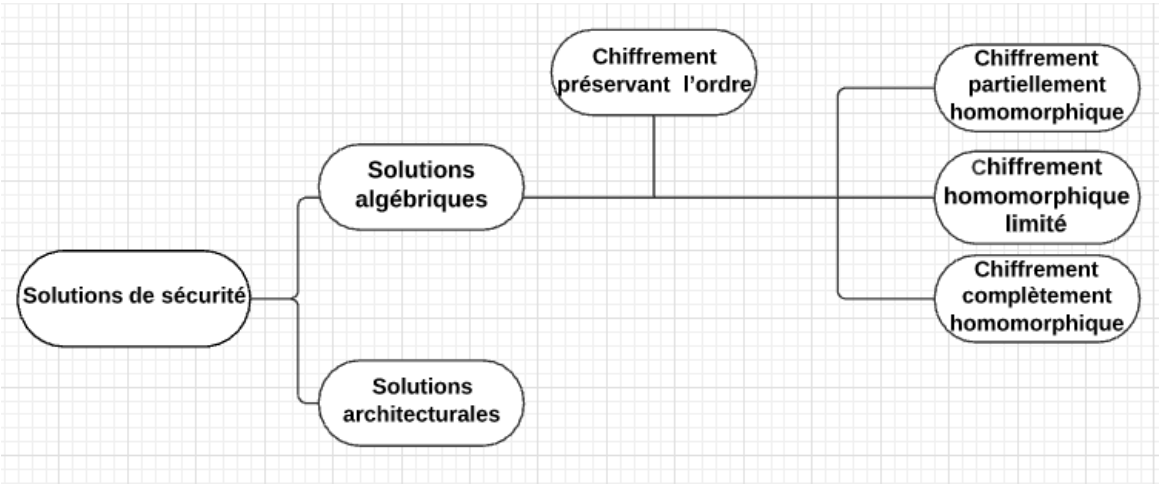


FIGURE 3.1 – les solutions de sécurité des données dans le Cloud.

3.4.1 Solutions architecturales

Des travaux récents [37][7][3][12] définissent des architectures de multi-Cloud permettant de maintenir la sécurité et d'atténuer les limites de sécurité dans le Cloud. Ces architectures sont adaptées aux plates-formes Cloud et n'introduisent pas généralement de proxy intermédiaire ou de serveur broker entre le client et le fournisseur de Cloud. L'utilisation de mécanisme multi-Cloud peut révéler diverses théories afin de cibler différents aspects de la sécurité de confidentialité, intégrité, consistance et cohérence de données stockées dans différents Clouds. Cette classe utilise le principe de distribution de manière systématique des données utilisateur sur plusieurs fournisseurs Cloud.

Le travail dans [36] fournit une construction de deux Clouds avec une série de protocoles de communication pour un service de base de données externalisé. La technique proposée garantie la protection et la confidentialité de données, les propriétés statistiques et la conception des requêtes. Le système proposé comprend un administrateur de base de données et deux Clouds non complotés. Dans ce modèle, l'administrateur de base de données peut être implémenté au côté du client. Les deux Clouds (notés Cloud A et Cloud B) fournissent le stockage et le service de calcul. Les deux Clouds travaillent ensemble pour répondre à chaque demande d'interrogation du client / des utilisateurs autorisés.

Comme le montre la figure 3.2, pour agir avec une base de données sécurisée, les données sont chiffrées et externalisées pour être stockées dans un Cloud (Cloud A) et les clés privées sont stockées dans l'autre (Cloud B). Les résultats

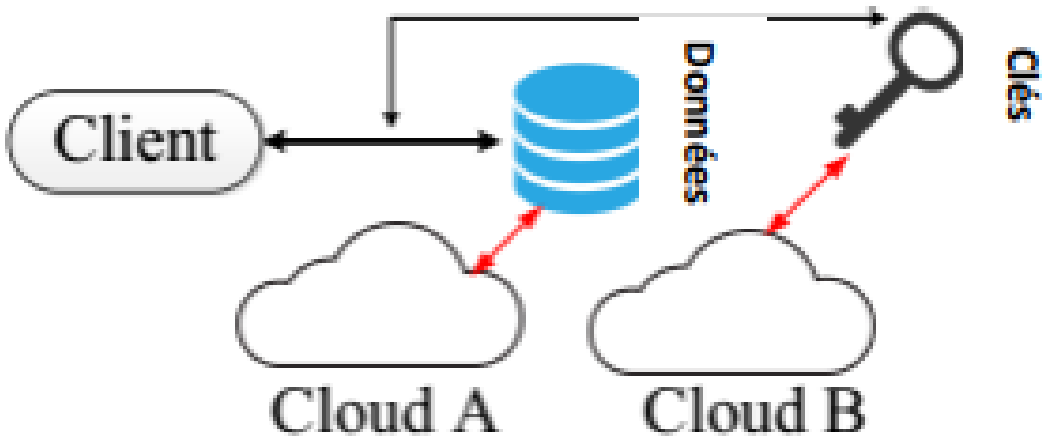


FIGURE 3.2 – Architecture avec deux Clouds.

expérimentaux démontrent que l'approche répond aux exigences de confidentialité et la méthode proposée est efficace dans les bases de données stables, tandis que l'efficacité de l'approche diminue dès que le volume de bases de données augmente. Dans ce cas la complexité de l'approche est plus élevée et l'espace mémoire nécessaire est très augmenté et le traitement des requetés ne couvre pas certaines opérations telles que l'opération Somme.

Les auteurs dans [17] ont proposé un schéma de base de données en tant que service (SecureDBaaS) dans lequel les données sont conservées dans un état confidentiel et sécurisé. Ce travail prend en charge l'exécution des opérations simultanées et indépendantes sur une base de données chiffrée dans le Cloud. SecureDBaaS implémente de nombreuses techniques pour convertir les données de texte en clair vers données, métadonnées ou structures de données chiffrées. Dans SecureDBaaS, il existe principalement trois opérations dont la première est la phase d'installation, dans laquelle la table de stockage de métadonnées est créée. Cette métadonnée contient les types de données, les clés principales ainsi que les techniques de chiffrement. La deuxième phase est la phase de l'utilisation qui consiste à générer les requêtes SQL exécutable et qui ont lieu après l'authentification d'un utilisateur. Un client de SecureDBaaS analyse l'opération et récupère les métadonnées pour réaliser la communication. Les informations des métadonnées sont utilisées pour convertir le SQL en clair vers la requête exécutable dans le Cloud. Enfin, dans la phase finale, les résultats sont ensuite déchiffrés et remis à l'utilisateur. Cette approche est basée sur l'exécution d'opérations de chiffrement via des algorithmes de chiffrement SQL-aware, qui avait été initialement proposée dans [4]. Il s'agit donc des techniques de chiffrement coûteux, et la table de stockage des métadonnées est située sur un serveur non

confiant, ce qui pose d'énormes problèmes pour la sécurité de ce schéma.

3.4.2 Solutions algébriques

Les limitations, essentiellement la confidentialité, obligent les chercheurs à proposer d'autres techniques de protection de données. Certains travaux exploitent un ou plusieurs techniques de chiffrement en Cloud et par conséquent la notion la plus connue « Homomorphique ». Dans les techniques proposées, nous pouvons étudier les cryptosystèmes de chiffrement partiellement homomorphique, chiffrement homomorphique limité et chiffrement complètement homomorphique. Le homomorphique est une propriété qui permet de confier des calculs dans le Cloud, sans que les données ni les résultats soient accessibles au fournisseur de Cloud.

3.5 chiffrement homomorphe

3.5.1 chiffrement homomorphe additif

Dans un contexte de chiffrement additif, un serveur distant pourra retourner le résultat d'une opération d'addition sur les messages en clair en faisant le calcul sur des messages chiffrées, sans disposer de la clé secrète.

Définition 3.2 :
Un chiffrement homomorphe est additif si :

$$Enc(x \otimes y) = Enc(x) \oplus Enc(y)$$

$$\prod_{i=1}^{n} Enc(m_i) = Enc(\sum_{i=1}^{n} m_i)$$

En d'autres termes, soient :

Enc_p une fonction de chiffrement à clé publique p.

Dec_s une fonction de déchiffrement à clé secrète s.

Alors :

$$Dec_s(Enc_p(\mathrm{m}) \times Enc_p(\mathrm{n})) = \mathrm{m} + \mathrm{n}$$

Les chiffrements qui réalisent cette propriété de chiffrement Homomorphe additif sont : Paillier et Goldwasser-Micalli.

3.5.1.1 le chiffrement Homomorphe de Paillier

Le cryptosystème de Paillier est un cryptosystème asymétrique, conçu par Pascal Paillier en 1999. Ce cryptosystème est celui qui a la plus grande bande passante, appelée aussi taux d'expansion : rapport entre la longueur du clair et la longueur du chiffré [27]. Ce cryptosystème est basé sur les propriétés de la fonction lambda de Carmichael dans Z .

Génération des clés : - On choisit deux grands nombres premiers p et q ;

- On Calcule n = pq ;

- On choisit un entier g $\epsilon \, \mathrm{Z}_{n^2}^*$ tel que n et L($g^\lambda \mod n^2$) sont premiers entre eux, où L désigne la fonction :

$$\mathrm{L} : {Z_{n^2}}^* \longrightarrow \mathrm{Z}_n$$

$$\mathrm{u} \mapsto \frac{u-1}{n}$$

λ désigne la fonction de Carmichael :$\lambda(p.q)$ = ppcm(p - 1, q - 1).
La clé publique est donc formée de (n,g) et la clé privée des deux facteurs premiers (p,q).

L'algorithme de Paillier est détaillé ci-dessous :

Cryptosystème de Paillier :

- **Génération des clés :**
  - Choisir p et q premiers
  - Calculer $n = pq$
  - Choisir $g \in \mathbb{Z}_{n^2}^*$ tel que :

  $ppcm(L(g^\lambda \mod n^2), n) = 1$ avec $L(u) = \dfrac{u-1}{n}$

  Clé publique : pk = (n,g)
  Clé privée : sk = (p,q)

- **Chiffrement : Enc (m,pk,r)**
  - Choisir $r \in \mathbb{Z}_n^*$
  - Calculer $c = g^m.r^n \mod n^2$

- **Déchiffrement : Dec (c,sk)**
  - Calculer $m = \dfrac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod n^2)} \mod n$

Supposons qu'on a c_1 et c_2 deux textes chiffrées avec l'algorithme de Paillier et m_1 et m_2 les textes clairs correspondants tel que :

$$c_1 = g^{m_1}.r_1^n \bmod n^2$$

$$c_2 = g^{m_2}.r_2^n \bmod n^2$$

Alors :

$$c_1.c_2 = g^{m_1}.r_1^n.g^{m_2}.r_2^n \bmod n^2$$

$$= g^{m_1+m_2}.(r_1 r_2)^n \bmod n^2$$

Donc, le chiffrement de Pailler réalise la propriété du chiffrement Homomorphe additif.

Exemple d'application : Vote électronique [18]

Soient : l'ensemble des Candidats = X, Y, Z et l'ensemble des Électeurs = V_1, V_2, ..., V_n

La Figure 3.3 montre que, pour chaque candidat, on chiffre les valeurs du vote de chaque électeur avec la clé publique du chiffrement homomorphe additif "Paillier", ensuite pour obtenir la somme des votes, on déchiffre le produit des votes chiffrés comme suit :

$$Dec_{s_k}(\Pi_{i=1}^n C_i, x) = Dec_{s_k}(Enc_{p_k}(Tot_x)) = Tot_x$$

Candidats / Électeurs	X	Y	Z
V_1	$C_{1,X} = \text{Enc}(1, pk_A)$	$C_{1,Y} = \text{Enc}(0, pk_A)$	$C_{1,Z} = \text{Enc}(0, pk_A)$
V_2	$C_{2,X} = \text{Enc}(1, pk_A)$	$C_{2,Y} = \text{Enc}(0, pk_A)$	$C_{2,Z} = \text{Enc}(0, pk_A)$
V_3	$C_{3,X} = \text{Enc}(0, pk_A)$	$C_{3,Y} = \text{Enc}(0, pk_A)$	$C_{3,Z} = \text{Enc}(1, pk_A)$
⋮	⋮	⋮	⋮
V_n	$C_{n,X} = \text{Enc}(0, pk_A)$	$C_{n,Y} = \text{Enc}(1, pk_A)$	$C_{n,Z} = \text{Enc}(0, pk_A)$
Total	?	?	?

FIGURE 3.3 – les votes des électeurs chiffrés un par un.

la figure 3.4 exprime le résultat obtenu après déchiffrement via la clé privée du chiffrement homomorphe de "Paillier" .

Électeurs \ Candidats	X	Y	Z
V_1	1	0	0
V_2	1	0	0
V_3	0	0	1
⋮	⋮	⋮	⋮
V_n	0	1	0
Total	Tot_X	Tot_Y	Tot_Z

FIGURE 3.4 – Application de "Paillier" pour retrouver la somme des votes

3.5.1.2 Le chiffrement Homomorphe de Goldwasser-Micalli

Le cryptosystème de Goldwasser-Micali (GM) est un algorithme asymétrique de cryptographie à clé publique, développé par Shafi Goldwasser et Silvio Micali en 1982. Goldwasser et Micali ont introduit la notion de chiffrement probabiliste, tout système de chiffrement doit intégrer de l'aléa dans le processus de chiffrement pour être considéré comme sûr. Le schéma de GM [20] qui repose sur la difficulté du problème de la résiduosité quadratique n'est pas efficace : les textes chiffrés peuvent être des centaines de fois plus longs que les textes d'origine.

Génération des clés : Le problème de résiduosité quadratique : étant donné un entier composite impair n, et un a ϵ $Z_n{}^*$ n tel que $(\frac{a}{n})$, décider si (a) est ou non un résidu quadratique modulo n.
Supposons n = pq (produit de deux nombres premiers), $(\frac{a}{n})$=1 implique soit $(\frac{a}{p})$=$(\frac{a}{q})$=1 (a est résidu quadratique) ou$(\frac{a}{p}) = (\frac{a}{q}) = -1$(a est non-résidu quadratique).

- On Choisit p et q premiers ;

- On Calcule n=pq ;

- On Choisit z ϵ Z_n tel que z soit un résidu non quadratique modulo n et $\frac{z}{n}$) =1

La clé publique est donc formée de (n,z) et la clé privée des deux facteurs premiers (p,q).

L'algorithme de Goldwasser-Micalli se présente comme suit :

Cryptosystème de Goldwasser-Micalli :

- **Génération des clés :**
  - Choisir p et q premiers
  - Calculer $n = pq$
  - Choisir $z \in \mathbb{Z}_n$ tel que : $(\frac{z}{n}) = 1$ et $(\frac{z}{p}) = -1$

  Clé publique : pk = (n,z)
  Clé privée : sk = (p,q)

- **Chiffrement : Enc (m_i,pk,r_i)**
  Soit $\mathcal{M} = \{0,1\}$ l'espace des messages en clair ;
  Pour tout $m \in \mathcal{M}$, m est composé de t bits : $m_1 m_2 ... m_t$
  - Choisir aléatoirement pour $\forall i \in [1,t]$ un r_i
  - Calculer $c_i = z^{m_i} . r_i^2 \mod n$

- **Déchiffrement : Dec (c_i,sk)**
  - Calculer $(\frac{c_i}{p}) = e_i$ pour $\forall i \in [1,t]$, avec $c = c_1 c_2 ... c_t$

  Si $e_i = 1$ alors $m_i = 0$; sinon $m_i = 1$

Le chiffrement de Goldwasser-Micalli (section 2.2.4) est un chiffrement XOR Homomorphe, qui n'a pas d'application concrète actuellement.

Supposons qu'on a c_1 et c_2 deux chiffrés de m_1 et m_2, on aura donc (Démonstration voir [11]) :

$Dec_{s_k}(Enc(p_k, m_1) x Enc(p_k, m_2)) = Dec_{s_k}(Enc(p_k(z^{m_1+m_2} . r_1^2 r_2^2) \mod n)$
$= m_1 \oplus m_2$

3.5.2 chiffrement homomorphe multiplicatif

Par analogie avec ce qui précède, un système basé sur le chiffrement homomorphe multiplicatif permet de n'effectuer que des produits sur les clairs, sans disposer de la clé secrète.

Définition 3.3 :
Un chiffrement homomorphe est multiplicatif si :
$Enc(x \otimes y) = Enc(x) \otimes Enc(y)$
$\prod_{i=1}^{n} Enc(m_i) = Enc(\prod_{i=1}^{n} m_i)$

En d'autres termes, soient :

Enc_p une fonction de chiffrement à clé publique p.

Dec_sune fonction de déchiffrement à clé secrète s.

Alors :

$$Dec_s(Enc_p(m) \text{ x } Enc_p(n)) = \text{m x n}$$

Parmi les algorithmes Homomorphes permettant ce type d'opération, nous citons RSA et El Gamal.

3.5.2.1 Le chiffrement Homomorphe de RSA

Le premier système à clé publique à être proposé fut celui de Ronald Rivest, Adi Shamir et Leonard Adleman connu sous le nom RSA. Cet algorithme a été décrit en 1978 [31]. Parmi tous les systèmes cryptographiques asymetriques à l'heure actuelle, RSA est considéré comme un des plus solides, si ce n'est le plus solide. Il a résisté à des années de cryptanalyse intensive et il est encore jugé assez robuste pour protéger les échanges bancaires et autres données critiques. Ce niveau de sécurité réside dans la difficulté de factoriser des grands nombres. Retrouver le texte en clair à partir d'une clé et du texte chiffré est supposé équivalent à la factorisation du produit des deux nombres premiers.

Les étapes de la génération des clés de chiffrement et déchiffrement de l'algorithme de RSA sont les suivantes :

Génération des clés : - On Choisit deux nombres premiers p et q et on calcule le produit n=pq ;

- On Choisit ensuite une clé de chiffrement aléatoire e, tel que e et (p-1)(q-1) soient premiers entre eux ;

- Finalement, on calcule la clé de déchiffrement de telle manière que :

d = e^{-1} mod ((p-1)(q-1))

La clé publique est donc formée des deux nombres e et n, la clé privée est le nombre d.

Ci-dessous l'algorithme de RSA en détail :

Cryptosystème de RSA :

- **Génération des clés :**
  - Choisir p et q
  - Calculer n=pq ; $\phi(n) = (p-1)(q-1)$
  - Déterminer d tel que : $e.d \equiv 1 \mod \phi(n)$

  Clé publique : pk = (e,n)

  Clé privée : sk = d
- **Chiffrement : Enc (m,pk)**
  - Calculer $c = m^e \mod n$
- **Déchiffrement : Dec (c,sk)**
  - Calculer $m = c^d \mod n$

Malgré sa robustesse, ce système s'avère vulnérable à l'attaque de l'homme du milieu, c'est à dire par interception et remplacement de la clé publique, l'attaquant récupère la clé publique d'un interlocuteur et fournit au second sa propre clé publique à la place.

Supposons qu'on a c_1 et c_2 deux textes chiffrées avec l'algorithme de RSA et m_1 et m_2 les textes clairs correspondants tel que :

$c_1 = m_1^e \mod$ n
$c_2 = m_2^e \mod$ n
$c_1.c_2 = m_1^e m_2^e \mod$ n
$= (m_1 m_2)^e \mod$ n

En déchiffrant le produit des chiffrés, on obtient le produit des clairs :

$Dec_s(c_1 c_2) = ((m_1 m_2)^e)^d \mod$ n
$= m_1 m_2$

Donc, le chiffrement de RSA réalise la propriété du chiffrement Homomorphe multiplicatif.

Exemple : Application numérique du chiffrement homomorphe multiplicatif de RSA :

Soit, pour p = 3, q = 5, e = 9 et d = 1 avec une taille de bloc = 4 ;

m_1 et m_2 deux messages clairs et c_1 et c_2 leurs chiffrés respectifs, obtenus en utilisant le chiffrement de RSA.

$m_1 = 589625 \rightarrow c_1 = 000500080009000600020005$
$m_2 = 236491 \rightarrow c_2 = 000200030006000400090001$

Après avoir chiffré les messages m_1 et m_2, la Figure 3.5 montre le résultat de la conversion binaire des chiffrés c_1 et c_2 de chaque bloc de 4, et la Figure 3.6 illustre le processus de multiplication de c_1 x c_2 bloc par bloc, ensuite la conversion du résultat de chaque multiplication en décimal.

00 05 => 00 0101	00 02 => 00 0010
00 05 => 00 0101	00 02 => 00 0010
00 08 => 00 1000	00 03 => 00 0011
00 09 => 00 1001	00 06 => 00 0110
00 06 => 00 0110	00 04 => 00 0100
00 02 => 00 0010	00 09 => 00 1001
00 05 => 00 0101	00 01=> 00 0001

FIGURE 3.5 – Conversion de c_1 et c_2 en binaire

00 0101×00 0010 = 00 1010	00 10
00 1000×00 0011 = 00 11000	00 24
00 1001×00 0110 = 00 110110	00 54
00 0110×00 0100 = 00 11000	00 24
00 0010×00 1001 = 00 10010	00 18
00 0101×00 0001 = 00 0101	00 05

FIGURE 3.6 – Multiplication de c_1 x c_2 bloc par bloc

c_1c_2 = 00100002000400050004000200040001000800005

En déchiffrant le chiffré c_1 x c_2 avec la clé privé de RSA, nous obtenons :

m_1m_2 = 102454241805

Vérification : m_1 = 589625 et m_2 = 236491

Nous multiplions m_1 x m_2 bloc par bloc, nous obtenons :

m_1m_2 = 10245424185

On remarque que c'est exactement le même résultat comme si le calcul a été mené sur les messages en clair : m_1 x m_2

3.5.2.2 Le chiffrement Homomorphe d'El Gamal

Le cryptosystème d'El Gamal est une méthode de cryptographie à clé publique inventée par Taher ElGamal en 1985. Sa sécurité repose sur la difficulté de calculer le logarithme discret [14].

La génération des clés de chiffrement et de déchiffrement pour le cryptosystème d'El Gamal se fait comme suit :

Génération des clés :

- On choisit un nombre premier p et deux nombres aléatoires g et x, tel que g et x soient inférieurs à p ;

- On calcule y = g^x mod p

La clé publique est donc formée de y, g et p, la clé privée est x.

L'algorithme d'El Gamal en détail :

Cryptosystème d'El Gamal :

- **Génération des clés :**
  - Choisir p premier, g et x aléatoires tel que (g,x < p)
  - Calculer $y = g^x \mod p$

  Clé publique : pk = (g,p)

  Clé privée : sk = x
- **Chiffrement : Enc (m,pk,K)**
  - Choisir un entier r
  - Calculer $K = g^r \mod p$
  - Calculer ensuite $c = my^r$
  - le message chiffré est c' = (K,c)
- **Déchiffrement : Dec (c,sk,K)**
  - Calculer $m = cK^{-x}$

Notons que ce calcul dépend du choix de K = g^r mod p (Voir l'algorithme ci-dessus), et donc pour un message clair donné, il y a plusieurs messages chiffrés correspondants, aussi pour chaque message chiffré il faut envoyer un second élément nécessaire au déchiffrement (K).

Supposons qu'on a c_1 et c_2 deux textes chiffrées avec l'algorithme d'El Gamal et m_1 et m_2 les textes clairs correspondants tel que :

$$c_1 = (m_1 y^r, g^r)$$

$$c_2 = (m_2 y^r, g^r)$$

$$c_1 c_2 = (m_1 m_2 . y^{2r}, g^{2r})$$

Le déchiffrement du produit se fait de la sorte :

$$Dec_s(c_1 c_2) = \frac{m_1 m_2 . y^{2r}}{g^{-2rx}}$$

$$= \frac{m_1 m_2 . g^{2xr}}{g^{-2xr}}$$

$$= m_1 m_2$$

On constate que le chiffrement d'El Gamal réalise aussi la propriété du chiffrement Homomorphe multiplicatif.

3.5.3 Chiffrement complètement Homomorphe

Contrairement au chiffrement partiellement homomorphe, avec le chiffrement complètement homomorphe nous pouvons réaliser tout type de calcul sur les données chiffrées stockées dans le Cloud sans les déchiffrer. L'application de ce chiffrement complètement Homomorphe constitue une brique importante dans la sécurité du Cloud. Plus généralement, on pourrait sous-traiter des calculs sur des données confidentielles à des serveurs situés dans Cloud tout en gardant la clé secrète qui permet de déchiffrer le résultat du calcul.

En 2014, le chiffrement homomorphe devient très prometteur : la commission européenne appelle dans son dernier appel à projet ICT à utiliser le chiffrement homomorphe dans des applications à l'horizon 2020. Le projet HEAT a réuni avec succès les chercheurs de pointe sur ce sujet en Europe (des universités de Leuven, Bristol et du Luxembourg) ainsi que des partenaires industriels spécialisés en cryptographie avancée (CryptoExperts, NXP et Thales) intéressés par le chiffrement homomorphe [15].

Définition 3.4 :
Un système de chiffrement complètement homomorphe est un cryptosystème permettant de faire des calculs sur les données chiffrées sans les déchiffrer. Formellement, si c_1 (respectivement c_2) est un chiffré de m_1 (respectivement m_2) il existe deux opérations $\square$ et $\circ$ telles que :
$Dec(c_1 \square c_2) = Dec(c_1) \circ Dec(c_2) = m_1 \circ m_2$

Le chiffrement complètement Homomorphe a été initié par Craig Gentry, ensuite DGHV une nouvelle version de son algorithme appliquée sur les entiers a vu le jour en 2010.

3.5.3.1 Chiffrement de Craig Gentry

La première construction d'un système complètement homomorphe a été décrite par Gentry en 2009 [19], où il utilise des idéaux d'anneaux de polynômes. La sécurité de ce schéma repose sur les réseaux idéaux

Pour chiffrer un message, l'idée est d'y ajouter du bruit, c'est-à-dire des petites erreurs. La clé secrète permet de supprimer ce bruit, à condition qu'il ne soit pas trop gros. Les opérations homomorphes qui sont effectuées impactent également ce bruit, les bruits vont grossir. On ne pourra déchiffrer le message que si les bruits initiaux sont choisis très petits. Pour dépasser cette limitation sur le nombre d'opérations, et lorsque le bruit devient trop important, Gentry applique la méthode de "bootstraping" ou d'amorçage.

Si le déchiffrement était suffisamment efficace, on pouvait alors changer de clé publique pour réduire le bruit. On commence par utiliser une première clé, puis, quand le bruit devient trop important, on utilise une seconde clé pour rechiffrer le même message.

Le bruit ou (l'erreur) e dans un idéal I d'un anneau R est définit par : e = kI ϵ I
subset R. Le message est alors chiffré en ajoutant ce bruit au message,

Enc(m) = c = m + kI

La procédure de déchiffrement consiste à retirer l'erreur. Les propriétés homomorphes du système sont réalisées, pour :

$c_1 = m_1 + k_1 I$

et $c_2 = m_2 + k_2 I$

on a :

$c_1 + c_2 = m_1 + m_2 + (k_1 + k_2)I$

et $c_1 \text{x} c_2 = m_1 \text{ x } m_2 + (m_1 k_2 + m_2 k_1 + k_1 k_2)I$

On peut déjà remarquer que le bruit est beaucoup plus affecté par une multiplication que par une addition. Approximativement, une addition double le bruit alors qu'une multiplication l'élève au carré. Si un trop grand nombre d'opérations est effectué, le bruit devient trop grand et la procédure de déchiffrement retourne un message erroné. Cependant, en évaluant régulièrement la procédure de déchiffrement de manière homomorphe, on peut éviter que cela arrive, et c'est exactement ce que fait le bootstrapping :

Etant donné un chiffré c de m, cette procédure retourne un chiffré c' de m où le bruit k' contenu dans c' est plus petit que le bruit k contenu dans c : $\|K'\| < \|K\|$

Cependant, pour pouvoir évaluer la fonction de déchiffrement de façon homomorphe, il est nécessaire que celle-ci soit suffisamment simple, ce qui n'est pas le cas initialement. Pour faire face à ce problème, Gentry réduit la complexité du circuit de déchiffrement en publiant un ensemble de vecteurs dont

la somme d'une partie d'entre eux est égale à la clé secrète. Ce problème est connu sous le nom de "[Sparse] Subset Sum Problem", et est prouvé NP-complet.

L'idée de Gentry est de partir d'un schéma dit "somewhat homomorphic encryption scheme" qui peut évaluer des additions et des multiplications tant que le bruit n'est pas trop grand, et de lui appliquer la procédure de bootstrap. Le schéma initial est basé sur le "Ideal Coset Problem". Cependant, pour appliquer la procédure de bootstrap en réduisant la complexité du circuit de déchiffrement il se base sur le "Sparse Subset Sum Problem".

3.5.3.2 Algorithme de DGHV

En 2010 Dijk, Gentry, Halevi et Vaikuntanathan [34]ont présenté un schéma complètement homomorphe (DGHV) qui est une application du chiffrement de Gentry sur les entiers et dont la sécurité repose sur le problème du diviseur commun approché.

Algorithme DGHV :
Génération des clés : r, p et q, $r \sim 2^n$, $p \sim 2^{n^2}$, $q \sim 2^{n^5}$, p premier
La clé privée : p
$Enc_{s_k}(m) = pq + 2r + m = c$
$Dec_{s_k}(c) = (pq + 2r + m \mod p) \mod 2$

exactitude : $pq \ggg 2r + m$
Ainsi : $c \mod p = 2r + m$
Donc : $(c \mod p) \mod 2 = (2r + m) \mod 2 = m$

Pour deux messages m_1 et m_2, soient c_1 et c_2 leurs chiffrés respectivement :

$$c_1 + c_1 = (q_1 + q_2)p + 2(r_1 + r_2) + m_1 + m_2$$

donc si :

$$2(r_1{+}r_2) + m_1 + m_2 \ll \mathrm{p}$$

Alors :

$$((c_1{+}c_2) \bmod \mathrm{p}) \bmod 2 = [2(\mathrm{r}_1 + r_2) + m_1 + m_2] mod 2$$

$$= m_1 + m_2$$

Ainsi, DGHV réalise la propriété du chiffrement homomorphe additif.

$$c_1 x c_2 = [q_1 q_2 p + (2r_1 + m_1) + (2r_2 + m_2)]p + 2(2r_1 r_2 + r_1 m_1 + r_2 m_1) + m_1 m_2$$

Donc si :

$$2(2r_1 r_2 + r_1 m_1 r_2 m_1) + m_1 m_2 \ll p$$

Alors :

$$((c_1 x c_2) \bmod \text{p}) \bmod 2 = [2(2r_1 r_2 + r_1 m_1 + r_2 m_1) + m_1 m_2] \bmod 2 = m_1 m_2$$

De ce fait, DGHV réalise aussi la propriété du chiffrement homomorphe Multiplicatif.

Exemple numérique simple (Addition et multiplication DGHV) :

Calculons : 5 + 5 = 10, pour p=3, q=5 et r=3

L'opération du chiffrement :

$5 = (101)_2$
Enc(1) = 5 x 3+2 x 3 + 1 = 22
Enc(0) = 5 x 3+2 x 3 + 0 = 21
Enc(1) = 5 x 3+2 x 3 + 1 = 22
Enc(5) + Enc(5) = 222122 + 222122= 444244

L'opération du déchiffrement :
Dec(44) = 44 mod 3 = 2
Dec(42) = 42 mod 3 = 0
Dec(44) = 44 mod 3 = 2

Dec(444244) = 202 = $(1010)_2$ = 10

		2	0	2
=		0	1	0
+		2	0	0
=		2	1	0
=	1	0	1	0

Calculons : 5 x 5 = 25, pour p=3, q=5 et r=3

L'opération du chiffrement :

$5 = (101)_2$

Enc(1) = 5 x 3+2 x 3 + 1 = 22
Enc(0) = 5 x 3+2 x 3 + 0 = 21
Enc(1) = 5 x 3+2 x 3 + 1 = 22

Enc(5) x Enc(5) = 222122 x 222122

La multiplication se fait de la sorte :

			22	21	22
×			22	21	22
			484	462	484
+		462	441	462	
+	484	462	484		
=	484	924	1409	924	484

Enc(5) x Enc(5) = 484 924 1409 924 484

L'opération du déchiffrement :
Dec(484) = 484 mod 3 = 1
Dec(924) = 924 mod 3 = 0
Dec(1409) = 1409 mod 3 = 2
Dec(924) = 924 mod 3 = 0
Dec(484) = 484 mod 3 = 1

Dec(484 924 1409 924 484) = 1 0 2 0 1 = $(11001)_2 = 25$

3.5.4 Chiffrement partiellement Homomorphe

Un chiffrement qui n'est pas complètement Homomomoprhe, est forcement partiellement Homomomorphe, dans la mesure où il peut permettre la réalisation que d'une seule opération à la fois sur des chiffrés tel que (l'addition ou la multiplication), ou bien il accepte plus d'une opération mais avec un nombre limité d'iterations (pas plus d'une seule addition ou multiplication ou pas plus d'un bit).

Nous avons déjà détaillé dans les sections précédentes les algorithmes homomorphes : RSA, El Gamal, Paillier, Goldwasser-Micalli, Gentry et DGHV, dans ce qui suit, nous allons détailler d'autres algorithmes partiellement homomorphes : Benaloh, Okamot-Uchiyama, Sander-Young-Yung et Schmidt Samoa-Takagi.

3.5.4.1 Chiffrement de Benaloh

L'algorithme de Benaloh est détaillé ci-dessous :

Algorithme Benaloh :

Génération des clés : r la taille des blocs, p et q deux grands nombres premiers tel que :
$r|p-1$, $pgcd(r, \frac{p-1}{r}) = 1$ et $pgcd(r, q-1) = 1$
- Calculer : $N = pq$;
- Choisir : $y \in \mathbb{Z}_N^*$ au hasard tel que : $y^{\frac{(p-1)(q-1)}{r}} \mod N \neq 1$

La clé publique : (y,r,N)
La clé privée : (p,q)

Chiffrement : pour $m \in \mathbb{Z}_r$
- Choisir : $u \in \mathbb{Z}_N^*$ au hasard ;
- Calculer : $c = y^m u^r \mod N$

Déchiffrement :
- On doit faire une recherche exhaustive pour trouver quel $i \in \{0, ..., r-1\}$ vérifie : $(y^{-1}c \mod N)^{\frac{(p-1)(q-1)}{r}} \mod N = 1$

Le chiffrement de Benaloh est homomorphe pour l'addition :

Dec_{sk} ((c_1 x c_2) mod N) = (m1 + m2) mod r

3.5.4.2 Chiffrement d'Okamot-Uchiyama

Ci-dessous l'algorithme détaillé d'Okamot-Uchiyama :

Okamot-Uchiyama :

Génération des clés :
- Choisir p et q deux nombres premiers de k bits ;
- Calculer $N = p^2 q$;
- Choisir $g \in \mathbb{Z}_N^*$ au hasard tel que :

$g^p \mod p^2 \neq 1$

La clé publique : (N,g,h,k)
La clé privée : (p,q)

Chiffrement : pour $m \in \{1, ..., 2^{k-1} - 1\}$
- Choisir : $r \in \mathbb{Z}_N^*$ au hasard ;
- Calculer : $c = g^m h^r \mod N$

Déchiffrement :

$$m = \frac{c^{p-1} \mod p^2}{g^{p-1} \mod p^2} \mod p$$

Le chiffrement de Okamot-Uchiyama est homomorphe pour l'addition :

Dec_{sk} ((c_1 x c_2) mod N)=(m1 + m2) mod N

3.5.4.3 Chiffrement de Sander-Young-Yung

L'algorithme de Sander-Young-Yung se présente ainsi :

Sander-Young-Yung :

Génération des clés : se base sur le même principe de l'algorithme de Goldwasser-Micalli (section 3.2.2)
- Choisir p et q deux grands nombres premiers ;
- Calculer $N = pq$;
- Choisir aléatoirement un non-résidu quadratique y tel que : $\frac{y}{N} = +1$

La clé publique : (N,y)
La clé privée : (p,q)

Chiffrement : pour chiffrer un message $m \in \{0, 1\}$
- Choisir un entier $l \geq 1$
  - si $m = 0$; choisir un vecteur aléatoire $m' \in (Z_2)^l \neq 0$
  - si $m = 1$; $m' = 0^l$
- Utiliser la fonction de chiffrement de Goldwasser-Micalli pour chiffrer les m_i pour tout $1 \leq i \leq l$

$c = (Enc_{GM}(b'_1), ..., Enc_{GM}(b'_l))$

Déchiffrement :
$\forall 1 \leq i \leq l$;
- $d_i = 1$ si c_i est un carré modulo N, 0 sinon
- Si $d = 0^l$ alors $m = 1$, sinon $m = 0$

Le chiffrement de Sander-Young-Yung est homomorphe pour la multiplication :

Dec_{sk} $((c_1 \otimes c_2)$ mod N$)$=$(m_1$ x $m_2)$ mod 2

Où $\otimes$ correspond à une multiplication coordonnée par coordonnée.

3.5.4.4 Chiffrement de Schmidt Samoa-Takagi

L'algorithme de Schmidt Samoa-Takagi en détail :

Schmidt Samoa-Takagi :

Génération des clés :
- Choisir p et q tel que $p \nmid q-1$ et $q \nmid p-1$;
- Choisir l tel que $2^l < pq < 2^{l+1}$;
- Calculer $d = N^{-1} \mod (p-1)(q-1)$

La clé publique : (N,l)
La clé privée : (p,q,d)

Chiffrement : pour chiffrer un message $m \in Z_{2^l}$
- Choisir un entier $r \in \mathbb{Z}_N^*$ au hasard ;
- Calculer $c = r^N(1+mN) \mod N^2$

Déchiffrement :
- Calculer $r = c^d \mod pq$;

- Calculer $\dfrac{(r^{-N}c \mod N^2) - 1}{N} \mod pq$

Le chiffrement de Schmidt Samoa-Takagi est homomorphe pour l'addition :

Dec_{sk} ((c_1 x c_2) mod N^2)= (m_1 + m_2) mod pq

aussi :

Dec_{sk} (c_1^k mod N^2) = km_1 mod pq

3.5.5 Chiffrement homomorphique limité

Le chiffrement homomorphique limité (SomeWhat Homomorphic Encryption SWHE) permet d'exécuter plusieurs multiplications et de plusieurs additions, mais le nombre et le mode des opérations est limité. En termes spécifiques, les systèmes SWHE permettent un nombre illimité d'additions et de multiplications sur des données chiffrées. Le cryptosystème [23] était le premier algorithme permettant de calculer avec des schémas SWHE sur des textes chiffrés. L'addition est autorisée plusieurs fois tandis que la multiplication n'est possible qu'une seule fois.
Un autre exemple de schémas SWHE est présenté dans [26]. Il permet de calculer des polynômes multivariés de degré **d**. Il s'agit d'un système de chiffrement complétement homomorphique à **d** multiplication, c'est-à-dire qu'il permet **d**

multiplications et un nombre illimité d'additions sur des données chiffrées.

Dans [13], une nouvelle technique de gestion de base de données CryptDB, est utilisée pour protéger la confidentialité des données et traiter les requêtes SQL de manière efficace. CryptDB vise à exécuter des requêtes sur une base de données chiffrée et qui est gérée par le fournisseur de Cloud, à l'instant de l'exécution de requêtes sur une base de données non chiffrée. L'architecture CryptDB comprend une partie proxy de base de données et une partie SGBD chiffrée dans le Cloud. La partie proxy est considéré comme un serveur confiant, et qui stocke la clé principale secrète et le schéma de la base de données. Le proxy est utilisé comme couche intermédiaire qui chiffre et déchiffre toutes les données et modifie tous les opérateurs des requêtes. Dans cette approche, les données sont chiffrées en couches, appelées oignon [4]. Le mot « oignon » fait référence à des couches de chiffrement qui se chevauchent comme les couvertures d'un oignon. Ces oignons ont différentes couches, chacune chiffrée à l'aide d'algorithmes et de clés différents. CryptDB pose un problème majeur, qui consiste à effectuer des calculs au niveau du serveur sur des données chiffrées pour différents objectifs, car les textes en clair sont chiffrés avec des clés différentes. La solution traditionnelle consiste à exécuter le calcul après le déchiffrement des données. CryptDB est beaucoup plus efficace mais ne peut pas prendre en charge la plupart des requêtes analytiques sur des données chiffrées. De plus, il est trop coûteux pour certains calculs, comme pour les agrégats à grande échelle.

Les auteurs dans [21][32] ont proposé un chiffrement noté dodrantencryption qui utilise un chiffrement déterministe limité en conjonction avec un chiffrement semi-homomorphique et des tables logarithmiques et anti-logarithmiques afin de développer l'ensemble des requêtes SQL numériques. Ils ont proposé une solution palliative pour le chiffrement homomorphique qui utilise une variante déterministe du schéma de chiffrement de Paillier [9] et les grandes tables. Ce travail a permis de calculer des sommes et des produits de sommes, mais pas toutes les requêtes SQL impliquant des valeurs numériques. En particulier, il ne peut pas comparer deux valeurs numériques chiffrées.

3.5.6 Chiffrement préservant l'ordre

La propriété du chiffrement préservant l'ordre garantit la relation d'ordre entre les éléments de données en fonction de leurs valeurs chiffrées, sans révéler les données elles-mêmes (c'est-àdire que l'ordre entre les valeurs de texte en clair est conservé après le chiffrement). Cela permet de créer des index sur des données chiffrées qui peuvent être utilisées pour des requêtes d'ordre. Il existe de nombreuses approches pour établir cette propriété, comme [8][25][24] qui

utilisent des fonctions linéaires et non linéaires pour indexer les données.

En conclusion, les techniques de sécurité homomorphiques peuvent être assurées par des mécanismes cryptographiques et mathématiques distribués. La sélection du mécanisme le plus approprié est l'une des parties vitales du système. Le mécanisme sélectionné doit répondre aux contraintes des utilisateurs telles que la taille des clés, la taille des données, le temps de traitement et les caractéristiques des données tel que l'ordre, la possibilité de réalisation des opérations arithmétiques, etc

Dans ce chapitre, nous proposons un chiffrement complètement homomorphique qui préserve l'ordres dans un schéma construit par des simples expressions modulaires et linéaires de la forme (a $*$ x + b) mod p. La forme des expressions est publique, mais les coefficients a, b et p sont gardés secrets (non connus par une troisième partie). Notre schéma de chiffrement et d'indexation est théoriquement sécurisé, puisqu'un attaquant ne peut pas obtenir suffisamment d'informations pour résoudre les équations linéaires à partir des valeurs d'entrée et des informations générées. La section suivante présente le détail de la proposition.

3.6 Simulation des algorithmes Homomorphes

3.6.1 matériel utilisé

Pour cette simulation, nous avons utilisé un dispositif avec les caractéristiques suivantes :

PC : HP EliteBook. Processeur : Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz, 2.20 GHz.

RAM 8,00 Go, / SSD : 256 GO
type du système : système d'exploitation 64 bits, processeur x64

3.6.2 Étude comparative des cryptosystèmes Homomorphes

Chiffrement	Année de l'implémentation	Inventé par	Type de propriété	Type d'opération pris en charge
RSA	1977	Rivest, Shamir, Adleman	Asymétrique	Multiplicatif
ELGAMAL	1985	Taher Elgamal	Asymétrique	Multiplicatif
PILIER	1999	Pascal Paillier	Asymétrique	Additif
GOLDWASSER-MICALI	1982	Shafi Goldwasser et Silvio	Asymétrique	Additif
BENALOH	1994	Josh (Cohen) Benaloh	Asymétrique	Un seul soit (additif ou multiplicatif)
Okamot-Uchiyama	1998	Tatsuaki Okamoto et Shigenori Uchiyama	Asymétrique	Additif
Sander-Young-Yung	1999	Tomas SANDER - Adam L. YOUNG - Moti YUNG	Asymétrique	Multiplicatif
Schmidt Samoa-Takagi	2005	Katja SCHMIDT-SAMOA - Tsuyoshi TAKAGI	Asymétrique	Additif
Gentry	2009	Craig Gentry	Asymétrique	Additif et multiplicatif
DGHV	2010	Dijk, Gentry, Halevi et Vaikuntanathan	Asymétrique	Additif et multiplicatif

3.7 Analyse des performances

L'analyse des performances de la méthode proposée se fait par comparaison avec Paillier, RSA et Elgamal en termes de temps de chiffrement et de temps de déchiffrement.

Paramètres d'évaluation : La performance est évaluée dépend des mesures suivantes :

3.7.1 Temps de cryptage

Le cryptage est la méthode par laquelle il protège les informations précieuses qui incluent des images, des fichiers ou des transactions en ligne d'un accès non

officiel. C'est le processus dans lequel il prend le texte brut et une clé produite arbitrairement et en utilisant des opérations mathématiques avec les deux, afin qu'il soit mélangé. Le temps nécessaire pour effectuer ce processus est le temps de cryptage.

3.7.2 Temps de décryptage

Le processus de décryptage convertit à nouveau les données rendues incompréhensibles par le cryptage en leur forme non cryptée. Le processus de décryptage peut se faire automatiquement ou manuellement. Cela peut également être accompli par un groupe de clés ou de mots de passe.

3.8 Représentation du temps de chiffrement et du temps de déchiffrement

Le tableau suivant montre le processus de temps de chiffrement (ms) et le processus de temps de déchiffrement de Paillier, RSA et Elgamal Algorithm avec leurs schémas homomorphes respectifs avec une taille de données fixe et une longueur de clé variable.

Type de L'algorithme / Taille du clé (bit)	Processus de cryptage				Processus de décryptage			
	256	512	1024	2048	256	512	1024	2048
Algorithme RSA	4	5	43	67	20	22	95	567
Algorithme Elgamal	18	40	156	1213	30	70	309	786
Algorithme de Paillier	32	74	391	2520	39	90	589	4381

FIGURE 3.7 – Temps de cryptage et processus de temps de décryptage (ms)

la figure 3.7 représente les quatre tailles différentes de clé et le temps de chiffrement correspondant pris par RSA, Elgamal et Paillier Algorithm en utilisant leurs opérations respectives de propriété de chiffrement homomorphe. En analysant la figure 3.7 , nous concluons que le temps pris par le RSA en utilisant le temps de chiffrement homomorphe de petites données est très inférieur lorsqu'il est obtenu par rapport aux autres algorithmes. Le temps de chiffrement pris par les autres algorithmes avec des tailles de clé différentes est également montré dans la figure 3.8.

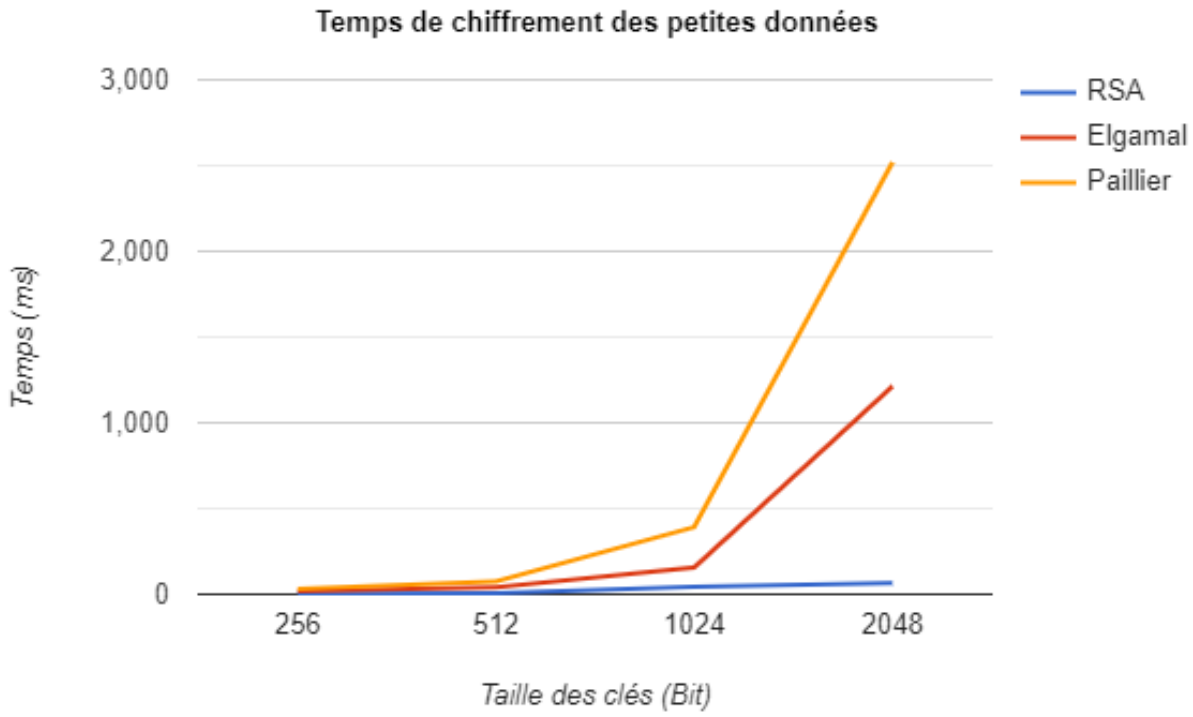


FIGURE 3.8 – Représentation du temps de cryptage

la figure 3.7 représente les quatre tailles différentes de clé et le temps de déchiffrement correspondant pris par RSA, Elgamal et Paillier Algorithm en utilisant leurs opérations respectives de propriété de chiffrement homomorphe. En analysant la figure 3.7, nous concluons que le temps pris par le RSA en utilisant le temps de cryptage homomorphe de petites données est très inférieur par rapport aux autres algorithmes. Ici, le temps pris par le Paillier atteint un temps de traitement élevé pour déchiffrer les petites données avec différentes tailles de clés. Le temps de décryptage pris par les autres algorithmes avec des tailles de clés différentes est également montré dans la figure 3.9.

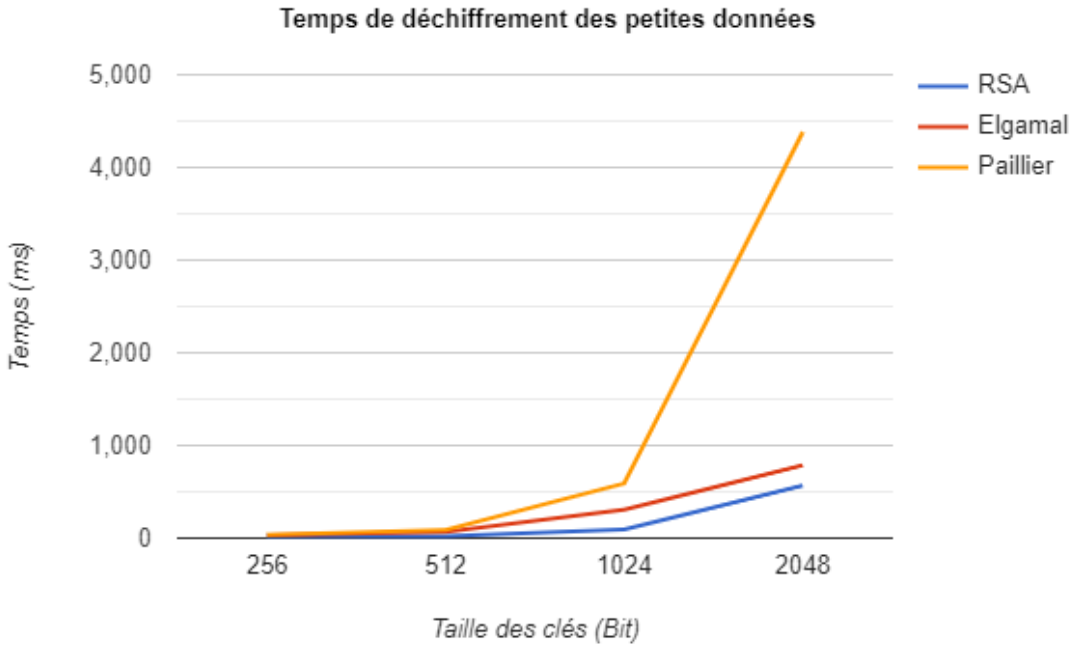


FIGURE 3.9 – Représentation du temps de déchiffrement

3.9 Chiffrement et déchiffrement de différentes tailles de données avec clé fixe

la figure 3.7 et la figure 3.10 suivante montrent le temps de cryptage et le temps de décryptage de Paillier, RSA et Elgamal Algorithm. De même, les figures 3.7 et 3.10 montrent le tracé du temps de cryptage, du temps de décryptage avec les différentes tailles de données de l'algorithme RSA Elgamal et Paillier. Le tableau 3.10 représente le temps de chiffrement et de déchiffrement de différentes données avec la taille de clé fixe de 2048.

Type d'algorithme & taille de données (bit)	Processus de cryptage		
	32B	1 Ko	10 Ko
Algorithme RSA	1	9	67
Algorithme Elgamal	43	1213	11847
Algorithme de Paillier	79	2520	22131

FIGURE 3.10 – Temps de cryptage (ms) de différentes tailles de données avec clé fixe

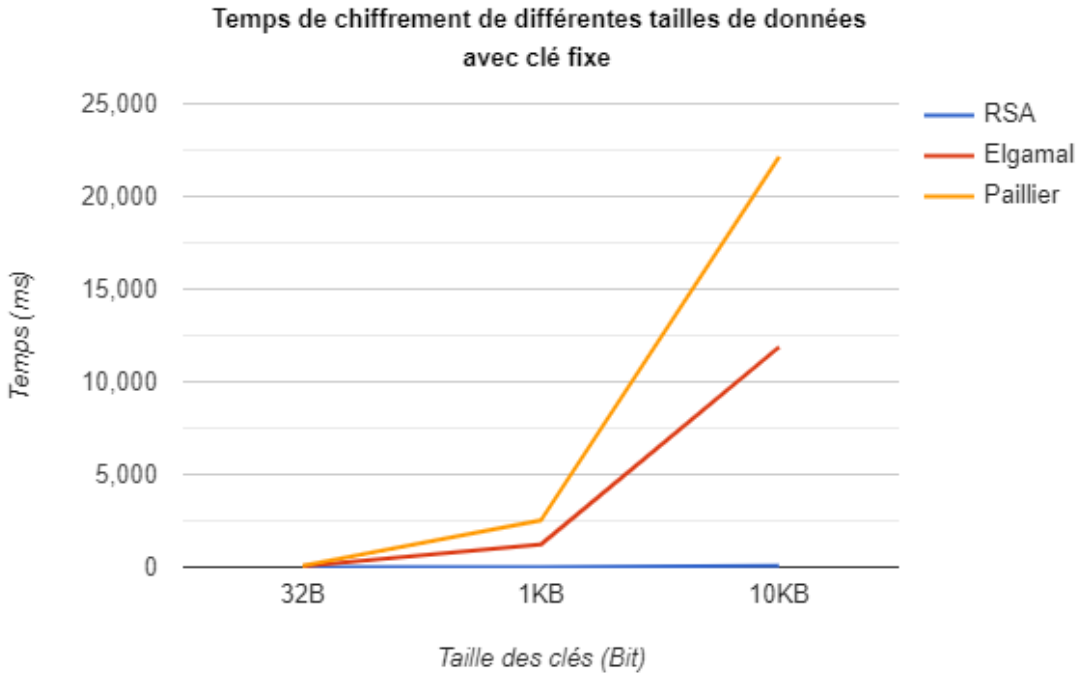


FIGURE 3.11 – Représentation du temps de chiffrement

La figure 3.11 montre le temps nécessaire à la multiplication Le temps de chiffrement de l'algorithme RSA, Elgamal et Paillier, où l'axe des abscisses représente les différentes tailles de clé et l'axe des ordonnées représente le temps en (ms) pris pour chiffrer les données par l'algorithme.

Type d'algorithme / **taille de données (bit)**	**Processus de décryptage**		
	32B	**1 Ko**	**10 Ko**
Algorithme RSA	25	567	5598
Algorithme Elgamal	19	786	6423
Algorithme de Paillier	12	4381	42129

FIGURE 3.12 – Temps de déchiffrement (ms) de différentes tailles de données avec clé fixe

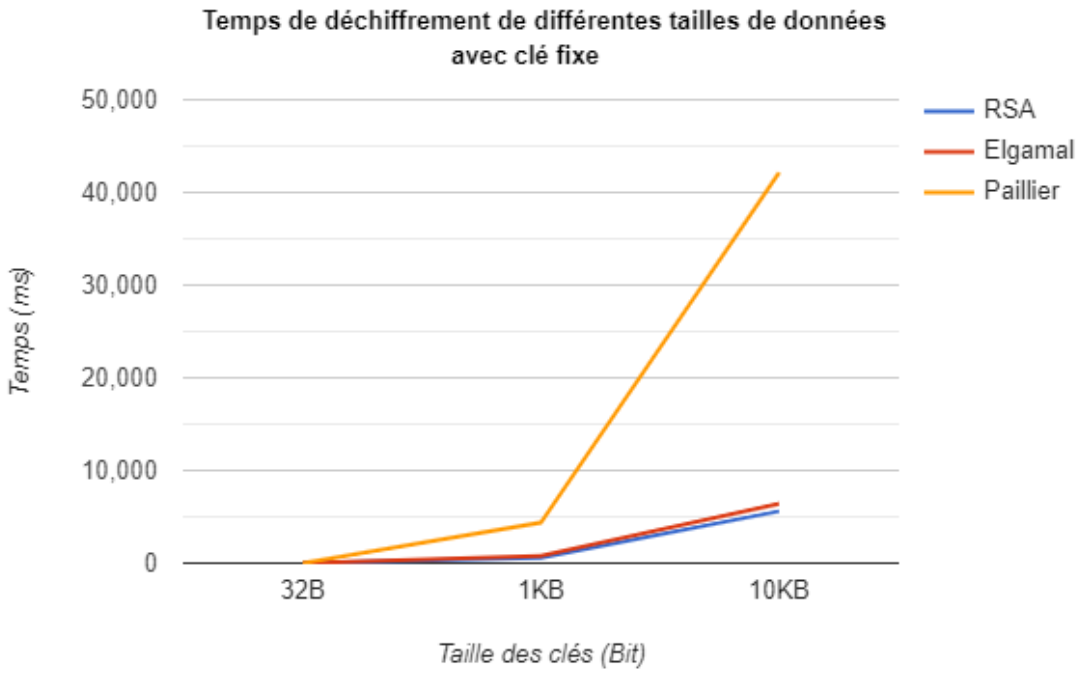


FIGURE 3.13 – Représentation du temps de déchiffrement

3.10 Comparaison des analyses des performances

La figure 3.8 montre le temps de cryptage des algorithmes RSA, Elgamal et Paillier, où l'axe des x représente la taille de la clé et l'axe des y représente le temps nécessaire pour crypter les données par l'algorithme.

Le résultat montre que le temps de traitement de l'algorithme RSA est inférieur à celui des algorithmes Paillier et Elgamal. Lorsque les données sont de 1 Ko, le temps de traitement avec les schémas précédents ne dépassera pas quelques millisecondes avec leur longueur de clé, ce qui est très inférieur aux bits de 1024. Mais avec les bits de 2048, le temps de traitement de l'algorithme de Paillier augmente avec le temps calculé de 3 secondes.

3.11 Conclusion

Le cryptage joue un rôle très important en termes de sécurité. Nous avons traité les performances des techniques de cryptage existantes comme les algorithmes RSA, Paillier et Elgamal. À partir du résultat de la simulation, nous avons évalué que l'algorithme RSA est bien meilleur que l'algorithme de Paillier et Elgamal. En conséquence, RSA est plus rapide lorsqu'il est comparé aux algorithmes d'Elgamal et de Paillier. L'Elgamal et le Paillier sont appelés algorithmes probabilistes mais, il est moins sécurisé que les systèmes de cryptographie précédents. Le RSA est dit algorithme déterministe

Conclusion générale et perspectives

Ce papier est une étude approfondie des différentes utilisations et implémentations du cryptage homomorphe afin d'améliorer les objectifs de sécurité et de confidentialité de tout système cryptographique. Dans un premier temps, nous avons présenté les travaux les plus pertinents de la littérature en mettant en évidence les principes du schéma adopté puis une brève description du modèle mathématique pour chaque travail. Le choix de la technique à utiliser dépend essentiellement des données elles-mêmes. De plus, pour obtenir le schéma le plus puissant, nous devons prendre en compte de nombreux paramètres : la taille du texte brut, la taille du texte chiffré, la taille de la clé, le temps passé dans chaque opération, soit le cryptage ou le décryptage, soit la génération de clé. À quel point notre système pourrait-il être sécurisé ? Pour répondre à cela, chaque aspect compte, et tant de paramètres doivent être bien ajustés, surtout quand il s'agit des données les plus sensibles.

Le cloud computing joue un rôle majeur dans la protection des données contre les attaques externes et les données sont protégées en utilisant plusieurs mécanismes de sécurité. L'objectif principal est de savoir comment stocker et activer les calculs effectués sur les données chiffrées en utilisant les propriétés de chiffrement homomorphes qui sont facilement disponibles dans l'environnement cloud. D'après les résultats, RSA serait plus rapide lorsque les algorithmes de Paillier et Elgamal sont considérés comme les plus efficaces en termes de sécurité.

En tant que travail futur, cette étude peut être étendue pour expérimenter d'autres schémas du chiffrement homomorphe, citant à titre d'exemple le chiffrement complètement homomorphe. Nous pensons aussi à une amélioration supplémentaire du domaine de cryptage homomorphe en facilitant autres opérations dans les schémas du chiffrement homomorphe à être effectuées sur des données cryptées. Cette recherche peut se concentrer sur l'optimisation et sur les tests d'algorithmes de cryptage homomorphes en stockant leurs données cryptées et peut être en mesure d'effectuer à la fois les opérations telles que l'addition et la multiplication des données stockées dans le réseau cloud.

3 Bibliographie

[1] A. chatterjee, i. sengupta, translating algorithms to handle fully homomorphic encrypted data on the cloud, ieee trans. cloud comput. 6 (1) (2018) 287–300.

[2] A. el-yahyaoui, m. el kettani, about fully homomorphic encryption improvement techniques, int. j. embedded real-time commun. syst. 10 (3) (2019) 1–20.

[3] A. m. a., p. e., s. b. et t. j. a., «cloud computing security : from single to multi-clouds,» system science (hicss), 2012 45th hawaii international conference on. ieee, p. 5490–5499, 2012.

[4] A. p. r., r. c., z. n. et b. h., «cryptdb : protecting confidentiality with encrypted query processing,» proceedings of the twenty-third acm symposium on operating systems principles. acm, p. 85–100, 2011.

[5] A. udmale, s.b. nimbekar, survey paper on digital image sharing by diverse image media, int. j. comput. appl. 137 (4) (2016).

[6] A.s. waleed, a.j. qussay, a.z. hani, cloud security based on the homomorphic encryption, int. j. adv. comput. sci. appl. 10 (10) (2019) 300—307.

[7] B. a., c. m., q. b., a. f. et s. p., «depsky : dependable and secure storage in a cloud-of-clouds,» acm transactions on storage (tos), vol. 9, n° %14, p. 12, 2013.

[8] B. a., c. n., l. y. et o. a., «orderpreserving symmetric encryption,» annual international conference on the theory and applications of cryptographic techniques. springer, p. 224–241, 2009.

[9] C. d., g. r., h.-g. n. et q. n. p., «paillier's cryptosystem revisited,» proceedings of the 8th acm conference on computer and communications security. acm, p. 206–214, 2001.

[10] C. gentry and s. halevi, n.p. smart, "homomorphic evaluation of the aes circuit", in proceeding of crypto, santa barbara, california, usa, pp :850-867, 2012.

[11] Clear m., hughes a. and tewari h. homomorphic encryption with access policies : Characterization and new constructions. in in africacrypt, pages 61–87, 2013.

[12] D. b. j., j. a. et o. a., «hail : A high-availability and integrity layer for cloud storage,» proceedings of the 16th acm conference on computer and communications security. acm, p. 187–198, 2009.

[13] D. z., «secure database in cloud computing-cryptdb revis-˘ ited,»international journal of information security science, vol. 3, n° %11, p. 129–147, 2014.

[14] El gamal t. a public key cryptosystem and a signature scheme based on discrete logarithms. advances in cryptology, springer berlin heidelberg, 1985.

[15] El gamal t. a public key cryptosystem and a signature scheme based on discrete logarithms. advances in cryptology, springer berlin heidelberg, 1985.

[16] F. armknecht, c. boyd, c. carr, k. gjøsteen, a. jaschke, c.a. reuter, m. strand, a guide to fully homomorphic encryption', cryptology eprint archive, report 2015/1192, 2015.

[17] F. l., c. m. et m. m., «distributed, concurrent, and independent access to encrypted cloud databases,» ieee transactions on parallel and distributed systems, vol. 25, n° %12, p. 437–446, 2014.

[18] Fabien l. and damien s. fully homomorphic encryption (fhe). equipe-projet inria du laboratoire de l'informatique du parallélisme, 2011.

[19] Gentry c. fully homomorphic encryption using ideal lattices. stoc, 9 :169–178, 2009.

[20] Goldwasser s. and micali s. probabilistic encryption. journal of computer and system sciences, 28(2) :270–299, 1984.

[21] J. s., l. w. et s. t., «numerical sql value expressions over encrypted cloud databases,» international conference on database and expert systems applications. springer, p. 455–478, 2015.

[22] J. singh, p. kaur, digital image watermarking of homomorphic encrypted images : A review, in : International conference on electrical, electronics, and optimization techniques (iceeot), 2016, p. 4.

[23] K. j., « theory of cryptography : Second theory of cryptography conference,» tcc 2005, cambridge, ma, usa, proceedings. springer, vol. 3378, 10-12 february 2005.

[24] L. d. et w. s., «nonlinear order preserving index for encrypted database query in service cloud environments,» concurrency and computation : Practice and experience, vol. 25, n° %113, p. 1967–1984, 2013.

[25] L. d. et w. s., «programmable order-preserving secure index for encrypted database query," in,» cloud computing (cloud), 2012 ieee 5th international conference on. ieee, p. 502–509, 2012.

[26] M. a., g. p. et h. j., «additively homomorphic encryption with d-operand multiplications,» annual cryptology conference. springer, p. 138–154, 2010.

[27] Paillier p. public-key cryptosystems based on composite degree residuosity classes. advances in cryptology eurocrypt, 1592 :223–238, 1999.

[28] P.v. parmar, s.b. padhar, s.n. patel, n.i. bhatt, r.h. jhaveri, survey of various homomorphic encryption algorithms and schemes, int. j. comput. appl. 91 (8) (2014) 26–32.

[29] R. busom. petrlic, f. sebé, c. sorge, m. valls, efficient smart metering based on homomorphic n encryption, comput. commun. j. 82 (15) (2016) 95–101.

[30] Renaud venet, url = "https ://www.renaudvenet.com/cloud-computing-avantages-et-inconvenients-2011-01-26.html", 26/jan/2011.

[31] Riverst r., shamir a. and adleman l. a method for obtaining digital signatures and public-key cryptosystems. communications of the acm, 21(2) :120–126, 1978.

[32] S. t., «dodrant-homomorphic encryption for cloud databases using table lookup,» networks, computers and communications (isncc), 2017 international symposium on. ieee, p. 1–6, 2017.

[33] url "https ://waytolearnx.com/2019/03/difference-entre-iaas-saas-et-paas.html" mars 18, 2019.

[34] Van dijk m., gentry c., halevi s. and vaikuntanathan v. fully homomorphic encryption over the integers. in advances in cryptology-eurocrypt, pages 24–43, 2010.

[35] "vaquero, l, rodero-merino, l, caceres, j and lindner m (2009). a break in the clouds : towards a cloud definition. acm sigcomm computer communications review. volume 39, issue 1, january 2009, pp 50-55 ".

[36] X. k., l. s., h. j., x. y., y. n. et h. p., «two-cloud secure database for numeric-related sql range queries with privacy preserving,» ieee transactions on information forensics and security, vol. 12,n° %17, p. 1596–1608, 2017.

[37] Z. o., d. a. et d. h., «cloud computing security through parallelizing fully homomorphic encryption applied to multi-cloud approach,» journal of theoretical applied information technology, vol. 87, n° %12, 2016.