F.S.D... N° D'ordre :....

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB BLIDA

Faculté des Sciences

Département d'Informatique

Mémoire présenté par : HAKEM Hadjer.

En vue d'obtenir le diplôme de Master

Domaine : *Mathématique et Informatique.* **Filière :** *Informatique.*

Spécialité : *Génie des logiciels.* **Option :** *Ingénierie de Logiciel.*

Une approche sécurisée de recherche d'informations sur des données cryptées du Cloud computing.

Encadreur: Mr F.Boucenna. **Promoteur : Mr A.H.Kameche.**

Soutenu le: 26/ 09/2016, devant le jury composé de :

Mr N.CHIKHI	**Président**
Mme S.AROUSSI	**Examinatrice**

Année Universitaire 2015/2016.

Abstract :

With the advent of Cloud Computing, data owners are motivated to outsource their complex data management systems to the public cloud for flexibility and savings. Although to protect the confidentiality of data, sensitive data should be encrypted before outsourced.

The encryption performed by the data owner, either storing plaintext and indicate sensitive data to the cloud service provider, or encrypt it, to prevent provider computation of that data. Both technique nor guarantee the performance or safety.

In order to meet these constraints, a new form of cryptography appeared, namely Homomorphic Encryption, which gives the possibility of analysis and processing on encrypted data without having to decipher them.

Our job is to study the homomorphic encryption and use it in the Information Retrieval Over Encrypted Data domain, and the implementation of a recent approach « Leveled Fully Homomorphyc Encryption », by exploitant the library HElib written in C++.

Key words : Cloud Computing, Homomorphic Encryption, Information Retrieval Over Encrypted Data

Résumé :

Avec l'avènement du Cloud Computing, les propriétaires de données sont motivés à externaliser leurs systèmes complexes de gestion de données au cloud public pour une grande flexibilité et des économies. Mais pour protéger la confidentialité des données, les données sensibles doivent être chiffrées avant l'externalisées.

Le chiffrement effectué par les propriétaires de données, soit stocker les données en clair et indiquer les données sensibles au fournisseur de services cloud, ou chiffrez les et empêcher le fournisseur de faire des opérations sur ces données. Les deux techniques ne garantient ni la performance ni la sécurité.

Afin de répondre à ces contraintes, une nouvelle forme de cryptographie est apparue, à savoir « le Chiffrement Homomorphe » qui donne la possibilité de faire des analyses et des traitements sur les données chiffrées sans avoir à les déchiffrer.

Notre travail consiste à étudier le chiffrement homomorphe et l'appliquer dans le domaine de la Recherche d'Information Crypté, ainsi que l'implémentation d'une approche récente « Leveled Fully Homomorphyc Encryption », en exploitant la librairie HElib écrite en C++.

Mots-clés: Cloud Computing, Chiffrement Homomorphe, Recherche d'Information Crypté.

Remerciement :

Tout d'abord, je remercie le bon Dieu, notre créateur de m'avoir donné la force, la volonté et le courage afin d'accomplir ce modeste travail.

Je tiens à remercier chaleureusement mes encadreurs au sein de CERIST **Mr O.NOUALI** et **Mr F.BOUCENNA** et mon promoteur au sein de l'université SDB **Mr A.KAMECHE** pour leurs nombreux conseils, encouragements, relectures, corrections et surtout pour leurs disponibilités et la confiance qu'ils m'ont accordé.

Mes remerciements, s'adressent également à monsieur le président du jury et les membres des jurys, pour l'honneur qu'ils me font d'assister à ma soutenance.

Je remercie l'ensemble des enseignants et corps administratif du département d'Informatique, qui m'a accompagnée durant mon cursus académique.

Pour finir, je remercie tous ceux qui ont contribué de prés ou de loin à l'aboutissement de ce travail.

Table des matières

Liste des figures

Liste de tableau :

Introduction Générale :

Aujourd'hui, l'information joue un rôle primordial dans le quotidien des individus et dans l'essor des entreprises. Cependant, le développement du Cloud et la généralisation de l'informatique dans tous les domaines ont conduit à la production d'un volume d'information sans précédent.

L'ère du Cloud Computing est arrivée! Cette innovation révolutionnaire comprend une puissance informatique et des capacités de stockage formidables sans posséder d'infrastructures ou de logiciels. En effet, l'infrastructure technique est « dans les nuages », cela signifie que les calculs et le stockage ont lieu sur des serveurs et des data-centers distants. Ainsi, le cloud offre l'opportunité aux entreprises, d'externaliser leurs données et la gestion de ces données.

Cependant l'externalisation de données sensibles est un grand risque de sécurité et de confidentialité substantiel. Les services Cloud récents offrent une protection fine de données avec un chiffrement de la source et à une authentification renforcée, mais reste toujours insuffisant d'utiliser une protection à base firewalls.

Le chiffrement est un ensemble de technologies matures qui peut être appliqué à une solution infonuagique afin de renforcer les contrôles de sécurité et de confidentialité des données.

Les méthodes de chiffrement varient, l'une des solutions proposée est de crypter les données localement avant l'externalisation en nuage, mais les données restent inexploitables. A moins de télécharger tous les données les déchiffrées, puis appliquer les traitements voulus, ce qui rend la récupération des données une tâche très difficile, vu l'énorme masse de données externalisée et la fréquence élever des traitements sur le cloud, cette solution est irréalisable.

Afin de répondre à cette problématique et corriger les faiblesses du chiffrement classique une nouvelle forme de cryptage a vu le jour, à savoir « chiffrement homomorphique », qui est en cour de développement, il offre une meilleure sécurité et permet la manipulation des données chiffrées, sans avoir à les déchiffrer.

C'est dans ce cadre s'inscrit notre projet de fin d'étude, pour lequel nous avons fixés, comme objectifs, l'étude du chiffrement homomorphique et son application dans le domaine de la recherche d'information sur les données cryptées.

Suivant cet objectif, ce mémoire est construit autour de deux parties, la première partie dénommé « Etude Bibliographique» comprend trois chapitre :

Chapitre1 : Présente un bref descriptif sur la recherche d'informations classique, son système, décrit aussi les différents modèles en particuliers le modèle booléen, le modèle vectoriel, et le modèle probabiliste.

Chapitre 2 : Introduit en détail le chiffrement homomorphique, en vigueur un historique de son évolution et les principales approches les plus dignes d'intérêt parmi le flot des méthodes existantes.

Chapitre 3 : Traite d'abord, la notion du chiffrement de données ou « la cryptographie », le but de la cryptographie et les principaux concepts cryptographiques. Nous abordons en détail le processus de la recherche d'informations sur les données cryptées, et analyser quelque méthodes appliquées dans ce domaine en citant les avantages et les inconvénients chacune de ces méthodes.

La deuxième partie comprend les chapitres quatre et cinq :

Chapitre4 : Est consacré à notre contribution qui consiste à proposer un modèle de la recherche d'informations cryptées en employant le chiffrement Homomorphique.

Chapitre5 : Le dernier chapitre présente l'implémentation de notre contribution par le développement d'un ensemble de programmes en exploitant la librairie HElib qui est l'implémentation la plus récente du chiffrement Homomorphique.

Nous concluons notre mémoire par une conclusion générale, où nous présentons les perspectives de nos propositions.

Partie I :

Etude Bibliographique.

Chapitre 1: Introduction à la Recherche d'Informations.

Introduction :

La Recherche d'Informations (RI) peut être définie comme une dicipline utilise des méthodes qui permettent d'organiser un volume de données et de situer l'information qui répond aux besoin exprimés par l'utilisateur. L'opération de la RI est réalisée par des outils informatiques appelés Systèmes de Recherche d'Informations (SRI). Ces systèmes ont pour but de mettre en correspondance une représentation du besoin de l'utilisateur (requête) avec une représentation du contenu des documents (index) au moyen d'une fonction de comparaison (ou de correspondance).

1. Définition :

Plusieurs définitions de la recherche d'informations ont vu le jour dans ces dernières années, nous citons dans ce contexte les trois définitions suivantes [1]:

- La recherche d'informations est une activité dont la finalité est de localiser et de délivrer des granules documentaires à un utilisateur en fonction de son besoin en informations.
- La recherche d'informations est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information.
- La recherche d'informations est une discipline de recherche qui intègre des modèles et des techniques dont le but est de faciliter l'accès à l'information pertinente pour un utilisateur ayant un besoin en information.

Toutes ces définition partagent l'idée que la RI a pour objet d'extraire d'un document ou d'un ensemble de documents(corpus) les informations pertinentes qui reflètent un besoin d'informations.

2. Les concepts de bases :

La recherche d'informations est considérée comme l'ensemble des techniques permettant de sélectionner à partir d'une collection de documents, ceux qui sont susceptibles de répondre aux besoins de l'utilisateur. La gestion de ces informations implique le stockage, la recherche et l'exploration des documents pertinents. De ce contexte plusieurs concepts clés peuvent être définis, nous avons donc trouvé utile de les clarifier [2]:

temps une forte probabilité d'être pertinents, et une faible probabilité d'être non pertinents. Etant donné une requête utilisateur Q et un document D, il s'agit de calculer la probabilité de pertinence du document pour cette requête. Deux possibilités se présentent: D est pertinent pour q et D n'est pas pertinent pour q. Les documents et les requêtes sont représentés par des vecteurs booléens dans un espace à n dimensions. Un exemple de représentation d'un document dj et une requête q est le suivant :

dj = (w1, j, w2, j, w3, j, .., wn, j),

q = (w1, q, w2, q, w3, q, .., wn, q). Avec wk, j ∈ [0 1] et wk, q ∈ [0 1].

La valeur de wk, j (resp. wk, q) indique si le terme t_k apparaît ou non dans le document dj (resp. q).

Le modèle probabiliste évalue la pertinence du document dj pour la requête q. Un document est sélectionné si la probabilité que le document d soit pertinent, notée p(R/D), est supérieure à la probabilité que d soit non pertinent pour q, notée p(R/D) où R est l'événement de pertinence et est l'événement de non pertinence. Le score d'appariement entre le document d et la requête Q.

Ces probabilités sont estimées par de probabilités conditionnelles selon qu'un terme de la requête est présent, dans un document pertinent ou dans un document non pertinent. Cette mesure de similarité entre la requête et les documents peut se calculer par différentes formules. Ce modèle a donné lieu à de nombreuses extensions [8].

Conclusion :

L'objectif de tout système de la recherche d'informations est de trouver pour un utilisateur l'information pertinente qu'il cherche dans une masse d'informations considérables. La forme la plus simple d'un SRI est celle qui permet à l'utilisateur de formuler sa requête, de traiter cette dernière et de lui fournir une liste de documents ou de références aux documents pertinents.

Pour des raisons de sécurité, les technologies de la recherche d'informations passent à un haut niveau de sécurité afin de protéger la vie privée de l'utilisateur vu que des informations sensibles telles que des e-mails et des données financières sont entretenus dans des centres de données(ou sur le cloud). Des approches ont été proposées afin de traiter et trouver des solutions pour assurer la sécurité, étudier dans le chapitre suivant.

La pondération locale permet de mesurer l'importance du terme. Elle prend en compte les informations locales du terme qui ne dépendent que du document. Elle correspond en général à une fonction de la fréquence d'occurrence du terme dans le document noté *tf* (*t*erm *f*requency) exprimer ainsi :

$$tf_{ij} = 1 + log\left(f(t_i, d_j)\right)$$

Où : $f(t_i,d_j)$ est la fréquence du terme t_i dans le document d_j.

Quant à la pondération globale, elle prend en compte les informations concernant le terme dans la collection.Un poids plus important doit être assigné aux termes qui apparaissent moins fréquemment dans la collection. Car les termes qui apparaissent dans de nombreux documents de la collection ne permettent pas de distinguer les documents pertinents des documents non pertinents. Un facteur de pondération globale est alors introduit. Ce facteur nommé *idf* (*i*nverted *d*ocument *f*requency), dépend d'une maniére inverse de la fréquence en document du terme et exprimé comme suit :

$$idf = log\left(\frac{N}{n_i}\right)$$

Où : n_i est la fréquence en document du terme considéré, N est le nombre total de documents dans la collection.

Les fonctions de pondération combinant la pondération locale et globale sont référencées sous le nom de la mesure *tf*×*idf* . Cette mesure donne une bonne approximation de l'importance du terme dans la collections de documents de taillle homogène. Cependant un facteur important est ignoré, la taille du document. En effet, la mesure (*tf*×*idf*), ainsi définie favorise les documents long, car ils ont tendance à répéter le même terme, ce qui accroit leurs fréquence, par conséquent augmentent la similarité de ces documents, vis-à-vis de la requête. Pour remédier à ce problème, des travaux ont proposé d'intégrer la taille du document dans les formules de pondération, comme facteur de normalisation.

Le modèle vectoriel caractérisé par sa prise en compte du poids des termes dans le docments, permet de retrouver des documents qui répondent partiellement à une requête.

4.3. Le modèle probabiliste :

Ce modèle est fondé sur le calcul de la probabilité de pertinence d'un document pour une requête. Le principe de base consiste à retrouver des documents qui ont en même

- ✓ F est le schéma du modèle théorique de représentation des documents et des requêtes
- ✓ R (q, d) est la fonction de pertinence du document d à la requête q Nous présentons dans la suite les principaux modèles de RI : le modèle booléen, le modèle vectoriel et le modèle probabiliste.

4.1. Le modèle booléen :

Les premiers SRI développés sont basés sur le modèle booléen, même aujourd'hui beaucoup de systèmes commerciaux(moteurs de recherche) utilisent le modèle booléen. Cela est dû à la simplicité et à la rapidité de sa mise en œuvre.

Le modèle booléen est basé sur la théorie des ensembles et l'algèbre de Boole. Dans ce modèle, un document d est représenté par un ensemble de mots-clés (termes) ou encore un vecteur booléen. La requête q de l'utilisateur est représentée par une expression logique, composée de termes reliés par des opérateurs logiques : ET ,OU, NON [8].

L'appariement (RSV) entre une requête et un document est un appariement exact, autrement dit si un document implique au sens logique la requête alors le document est pertinent. Sinon, il est considéré non pertinent. La correspondance entre document et requête est déterminée comme suit :

$$RSV(d,q) = \begin{cases} 1 & si\ d\ appartient\ à\ l'ensemble\ décrit\ par\ q \\ 0 & sinon \end{cases}$$

4.2. Le modèle vectoriel :

Dans ces modèles, la pertinence d'un document vis-à-vis d'une requête est définie par des mesures de distance dans un espace vectoriel. Le modèle vectoriel représente les documents et les requêtes par des vecteurs d'un espace à n dimensions, les dimensions étant constituées par les termes du vocabulaire d'indexation. L'index d'un document dj est le vecteur = (w1,j ,w2,j ,w3,j , ..,wn,j), où wk,j ∈ [0 1] dénote le poids du terme tk dans le document dj . Une requête est également représentée par un vecteur = (w1, q, w2, q, w3, q,.., wn, q), où wk, q ∈ [0 1] est le poids du terme tk dans la requête q.

Le modèle véctoriel offre des moyens pour la prise en compte du poids de terme dans le document. Dans la littérature, plusieurs schémas de pondérations ont été proposés. La majorité de ses schémas prennent en compte la pondération locale et globale.

modèles de RI classiques : le modèle booléen, le modèle vectoriel, le modèle probabiliste, le modèle connexionniste et le modèle de langue. Ces modèles sont détaillés dans [7].

3.5. La reformulation de requête :

Afin de rapprocher au mieux la pertinence système de la pertinence utilisateur, une étape de reformulation de la requête est souvent utilisée dans les systèmes de recherche d'informations. La reformulation de la requête consiste à modifier la requête de l'utilisateur par ajout de termes significatifs et/ou ré-estimation de leurs poids.

3.6. Le moteur de recherche :

C'est une machine spécifique (matérielle et/ou logicielle) capable de construire, sur la base d'une collection de documents, un index approprié, et ensuite offrir à l'utilisateur le moyen de rechercher dans cette collection. Après la saisie d'une requête, le moteur applique la fonction d'appariement entre la requête et les documents indexés, et retourne à l'utilisateur une liste de résultats.

La performance et l'efficience : la performance d'un SRI est sa capacité à satisfaire l'utilisateur en terme de pertinence des documents retournés. Nous parlerons de l'efficience d'un SRI lorsqu'il s'agit d'évaluer tous les critères autres que la performance, à savoir, le temps de réponse, le coût et le temps d'indexation, la facilité d'utilisation, la simplicité de l'interface utilisateur.

4. Les modèles de recherche d'informations :

Un modèle de RI a pour rôle de fournir une formalisation du processus de RI et un cadre théorique pour la modélisation de la mesure de pertinence. Il existe un grand nombre de modèles de RI textuelle développé dans la littérature. Ces modèles ont en commun le vocabulaire d'indexation basé sur le formalisme mots clés et diffèrent principalement par le modèle d'appariement requête-document.

Le vocabulaire d'indexation T = {ti}, i ∈ {1, ..., n} est constitué de n mots ou racines de mots qui apparaissent dans les documents. Un modèle de RI est défini par un quadruplet (D, Q, F, R (q, d)) : où

- ✓ D est l'ensemble de documents
- ✓ Q est l'ensemble de requêtes

afin de produire un ensemble de mots clés que le système pourra gérer aisément, pour pouvoir les utiliser dans le processus de recherche ultérieur. Cette opération est appelée indexation. Un index est une structure qui permet d'associer, à chaque terme (mot clés) d'indexation, la liste des documents qui contiennent ce terme. En plus de la simple relation d'appartenance d'un terme à un document, un index peut fournir d'autres informations, comme ; le poids du terme dans le document, la co-occurrence des termes dans un document et la position du terme dans le document. L'ensemble des termes extraits de tous les documents est stocké dans une structure spécifique appelée Fichier Inverse [4].

3.2. L'appariement requête/document :

Une fois l'indexation des requêtes et des documents est effectuée, l'appariement entre la requête et le document peut désormais s'effectuer. L'appariement consiste à détecter les documents pertinents pour la requête émise, et éventuellement calculer le degré de cette pertinence appelé aussi score ou RSV (Retrieval Status Value). Ce score sert à trier la liste des documents retournés.

3.3. La pondération des termes :

L'élément fondamental dans un système de recherche d'informations est la technique utilisée pour pondérer les termes [5] [6]. La pondération des termes est utilisée dans l'opération d'indexation, permettant d'associer à chaque terme son poids dans le document. La plupart des techniques de pondération des termes sont basées sur les facteurs Tf et Idf :

- ✓ Tf (term frequency) : cette mesure est proportionnelle à la fréquence du terme dans le document.
- ✓ Idf (Inverse of Document Frequency) : mesure l'importance d'un terme dans toute la collection. Un terme trop fréquent dans la collection ne doit pas avoir le même impact sur la collection qu'un terme moins fréquent.

3.4. Le modèle de recherche d'informations (MRI) :

Il englobe les notions d'index et de fonction d'appariement. La fonction d'appariement utilise des informations contenues dans l'index. Ce dernier doit donc être construit en rapport avec cette fonction. Le MRI prédit les documents qui sont pertinents (et calcule leur degré de pertinence) et ceux qui ne le sont pas. Il existe plusieurs

3. Systéme de recherche d'informations :

Un système de recherche d'informations (SRI) intègre un ensemble de modèles et de processus permettant de sélectionner des informations pertinentes en réponse au besoin d'un utilisateur exprimé à l'aide d'une requête [3]. Ces processus permettent :

- ✓ La représentation des documents et des besoins, ce processus est appelé indexation ;
- ✓ L'interrogation, la recherche et la sélection des informations répondant aux besoins d'un utilisateur, ce processus est appelé appariement requête-document ;
- ✓ La reformulation de la requête.

La figure1.1 illustre l'architecture d'un système de recherche d'informations.

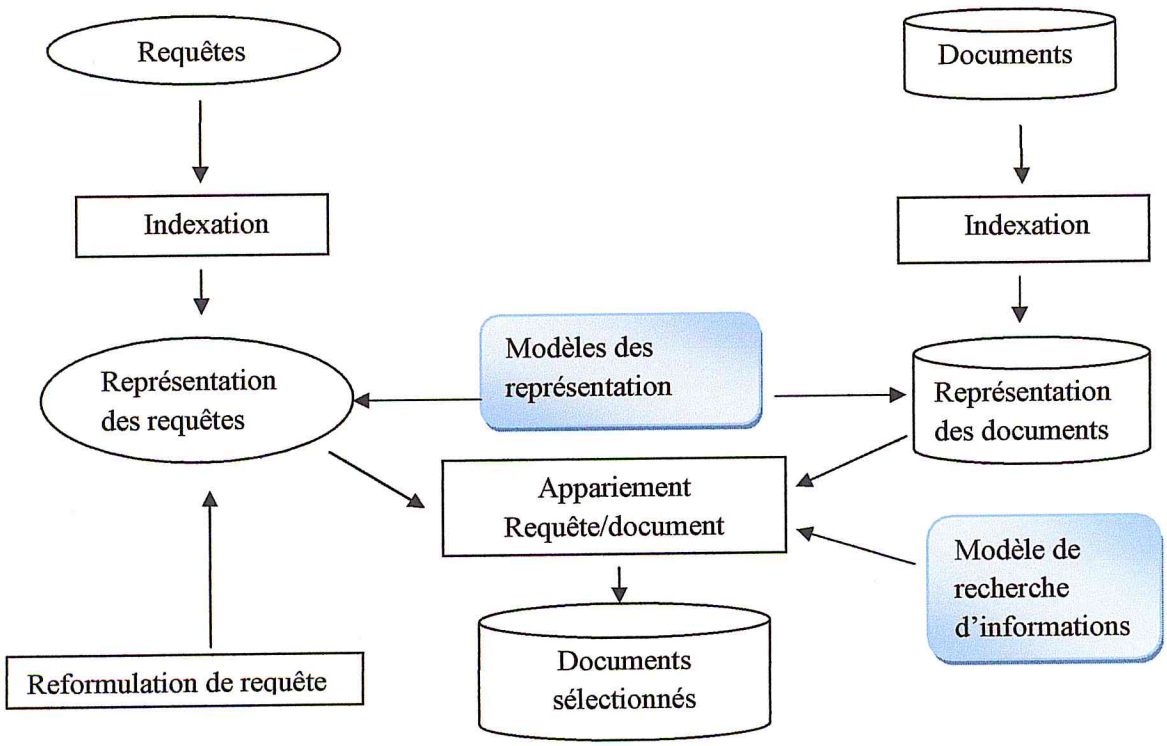

Figure1.1 : Architecture générale d'un systéme de recherche d'information[3].

Les composants de ce système sont décrits comme suit :

3.1. L'indexation :

Afin d'assurer la recherche dans des conditions acceptables en coût, une étape primordiale précédant l'étape de recherche effective d'informations doit s'effectuer. Celle-ci consiste à analyser le document lors de l'organisation du fonds documentaire

- ✓ **Collection de documents :** la collection de documents (ou fond documentaire) constitue l'ensemble des informations exploitables et accessibles. Elle est constituée d'un ensemble de documents.
- ✓ **Document :** le document constitue l'information élémentaire d'une collection de documents. L'information élémentaire, appelée aussi granule de document, peut représenter tout ou une partie d'un document.
- ✓ **Requête :** la requête constitue l'expression du besoin en informations de l'utilisateur. Elle représente l'interface entre le SRI et l'utilisateur. Divers types de langages d'interrogation sont proposés dans la littérature. Une requête est un ensemble de mots clés, mais elle peut être exprimée en langage naturel, booléen ou graphique.
- ✓ **Modèle de représentation :** un modèle de représentation est un processus permettant d'extraire d'un document ou d'une requête, une représentation paramétrée qui couvre au mieux son contenu sémantique. Ce processus de conversion est appelé indexation. Le résultat de l'indexation constitue le descripteur du document ou de la requête, qui est une liste de termes ou groupes de termes (concepts), significatifs pour l'unité textuelle correspondante, auxquels sont associés généralement des poids, pour différencier leurs degrés de représentativité du contenu sémantique de l'unité en question. L'ensemble des termes reconnus par le SRI est rangé dans une structure appelée dictionnaire.
- ✓ **Pertinence :** la pertinence est une notion fondamentale et cruciale dans le domaine de RI. Cependant la définition de cette notion complexe n'est pas simple, car elle fait intervenir plusieurs notions. Basiquement elle peut être définie comme la correspondance entre un document et une requête.
  - **Pertinence système :** c'est le degré de pertinence de l'information calculé par le système de RI.
  - **Pertinence utilisateur :** c'est le degré de pertinence de l'information correspondant au jugement de l'utilisateur.
- ✓ **Le bruit et le silence :** Le bruit, dans une réponse représente tous les documents non pertinents à la requête. Le silence, dans une réponse, représente tous les documents pertinents de la collection du système n'ayant pas été retrouvés pour cette requête.

Chapitre 2 :

Le chiffrement Homomorphique

Introduction :

Le cryptage homomorphique est une forme de chiffrement qui permet certains types de calculs à être effectués sur des données chiffrées et génère un résultat chiffré qui, une fois déchiffré correspond au résultat d'opérations effectuées sur les données en clair.

Dans ce présent chapitre nous commençons par un bref historique de cette récente discipline, puis nous définissons les types de chiffrement Homomorphique ensuite nous étudions les approches du HE qui existent déjà dans la littérature.

1. Historique :

En 1978, Ronald Rivest, Leonard Adleman et Michael Dertouzos suggèrent pour la première fois, le concept de chiffrement homomorphique [25]. Peu de progrès ont été accomplis depuis 30 ans. Le système de chiffrement de Shafi Goldwasser et Silvio Micali a été proposé en 1982. C'est un système de cryptage qui a atteint un niveau étonnant de sécurité. C'est un cryptage homomorphique additif, mais il peut crypter uniquement un seul bit.

Dans le même concept en 1999, Pascal Paillier a proposé un système de cryptage de sécurité qui était également un chiffrement homomorphique additif. Quelques années plus tard, en 2005, Dan Boneh, Eu-Jin Goh et Kobi Nissim [26] ont inventé un système de cryptage de sécurité, avec laquelle nous pouvons effectuer un nombre illimité d'addition, mais une seule multiplication.

En 2009, Craig Gentry d'IBM a proposé le premier système de cryptage "fully homomorphic Encryption" qui évalue un nombre arbitraire d'additions et de multiplications et donc calculer tout type de fonction sur des données chiffrées [27].

L'application de chiffrement entièrement homomorphique est une technologie importante dans la sécurité du Cloud Computing, de façon plus générale, nous pourrions confier les calculs sur des données confidentielles sur le serveur Cloud, en gardant la clé secrète qui peut décrypter le résultat du calcul.

IV. Homomorphic evaluation : l'algorithme $c_f \leftarrow$HE.Eval evk (f, $c_1,\ldots,c_\ell$) prend en entrée evk, une fonction f et un ensemble de ciphertexts $c_1,\ldots, c_\ell$ et en sortie ciphertext c_f.

3. Les Tentatives du chiffrement Homomorphique :

Nous étudierons dans cette section les récents schémas de chiffrement complètement homomorphique. Les différents systèmes ont été adoptés et chacun d'eux apporte une amélioration par rapport aux précédents. Le but est d'obtenir un crypto-système complètement homomorphique. Nous commencerons par le système de Craig Gentry:

A. Chiffrement complètement homomorphique (FHE), Craig Genry 2009 :

L'approche de Gentry [27] qui fût le premier crypto-système complètement homomorphe, son principe est de chiffrer des messages en leur ajoutant du bruit. Les opérations homomorphes effectuées influent sur ce bruit. L'approche de Gentry se base sur des idéaux d'anneaux de polynômes ou il choisit son bruit dans un idéal I d'un anneau R. Nous décrivons les algorithmes du système :

- keyGen : choisir un bruit dans un idéal I d'un anneau R et une clé K, et Erreur e = k I ou ($e \in I \subset R$)
- Enc(m, e) : le chiffrement du message est : c=m+e
- Dec(c,e) : le déchiffrement consiste à retirer l'erreur du message chiffré,

Les propriétés homomorphes du système : c1= m1+k1I et c2=m2+k2I

- ✓ Addition de c1 et c2 : c1+c2=m1+m2+ (k1+k2) I
- ✓ La multiplication de c1 et c2: c1.c2=m1.m2+ (m1k2+m2k1+k1k2) I

Le bruit est beaucoup plus élevé par une multiplication qu'une addition. Une addition double le bruit alors qu'une multiplication l'élève au carré. Si le nombre d'opérations effectuées est trop grand, le bruit devient important et la procédure de déchiffrement retourne un message erroné.

Pour éviter que cela arrive, Gentry applique une procédure dite "Bootstrap" (réamorçage ou réinitialisation) qui consiste à réduire ce bruit à un certain niveau permettant d'évaluer régulièrement la procédure de déchiffrement.

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.

B. Chiffrement complètement homomorphique sur les entiers (FHE Over Integers) 2010 :

A partir de la technique du chiffrement presque homomorphique "bootstrappable" SWHE, se base la construction du système fully homomorphic Encryption le chiffrement complètement homomorphiqe. Cependant, au lieu d'utiliser des idéaux d'anneaux de polynômes, ce schéma de chiffrement bootstrappable utilise simplement l'addition et la multiplication sur les entiers [32].

Ce schéma se diffère par rapport aux précédents selon deux caractéristiques : protection de circuit et le vrais défi dans cette construction est la compacité

La Construction du système est représentée par les procédures suivantes :

KeyGen :

- Clé secrète: grand nombre entier impair sk=n ;
- Clé publique: entiers, pk=(x_0,x_1,…,x_r) avec xi= nq_i+2r_i .

Encrypt: se base sur le sous-ensemble de somme des xi, m est le message à chiffré C= $(m+2r+\sum_i xi) \bmod x_0$.

Decrypt: m=(c mod n) mod 2

Add, Mult: faire la somme ou le produit sur les entiers.

Il se dit simple car il permet de réduire le temps de chiffrement et de déchiffrement, mais il comporte des failles de sécurité (les entiers proche d'un multiple d'un nombre caché, permettant de retrouver ce nombre) et de performance (ne supporte que des opérations sur des entiers ainsi qu'en pratique la grande taille des paramètres rend le système inapplicable).

C. Chiffrement complètement homomorphique base sur LWE (FHE from LWE) 2011:

Ce schéma de chiffrement complètement homomorphique (FHE) repose uniquement sur la méthode Learning With Error (LWE) [33], qui a été introduite par Regev en 2005. Son objectif est de résoudre un système d'équation linéaire. LWE fait intervenir trois paramètres : la dimension n, le module q, et le facteur d'erreur e. Les paramètres q et e sont souvent choisis comme des fonctions de n. La version calculatoire de LWE consiste à retrouver un vecteur s ϵ $\mathbb{Z}_q^n$ à partir d'un nombre

arbitraire de produits scalaires bruités entre s et des vecteurs connu a_i choisis uniformément dans $\mathbb{Z}_q^n$ à savoir <s, a1>+e_1, <s, a2 >+e_2, …, <s,a_n>+e_n

Sa version décisionnelle consiste à distinguer ces produits scalaires bruités de la distribution uniforme.

La construction du schéma améliore deux travaux précédents :

1. Ils montrent que "chiffrement presque homomorphic " (somewhatHE) est constitué, avec la nouvelle technique de re-linéarisation qui consiste à réduire la taille du message chiffré à n+1 au lieu de $n^2/2$ en utilisant la méthode LWE. Contrairement à, tous les schémas précédents basés sur des hypothèses liées aux idéaux dans divers anneaux.
2. Ils ont écartés le "squashing" utilisé dans tous les travaux précédents. Ils introduisent une nouvelle technique de réduction de dimension-module, qui transforme le schéma presque homomorphique (SWHE) en un schéma complètement homomorphique (FHE). L'idée est : considérons un message (m) chiffré avec un degré de polynôme maximum (n log q), faire convertir en un autre message chiffré pour le même message (m) par une modification des paramètres n et q en k et p respectivement qui sont plus petit, ce qui réduit la complexité de décryptage du système.

Brakarski et al décrivent leur système comme suit :

KeyGen :

- ✓ génération de la clé secrète sk= (s[1],…, s[n]) qui est un vecteur aléatoire,
- ✓ La clé publique pk= (A, v), A est une matrice aléatoire, e est un vecteur bruit ou v= Ask+2e.

Encryption : le chiffrement d'un message m ∈ {0, 1} s'effectue on choisissant un vecteur aléatoire e, $a=A^Te$ et $b=v^Te+m$

Decryption : en premier lieu, on recalcule le masque <a, sk> et on le soustrait de b, ce qui aura pour résultat : 2e+m (mod q), comme e<<q, on obtient donc 2e+m.

Evaluate : les propriétés homomorphique du shéma :

Soit un cryptogramme (a, b), et considérons la fonction linéaire :

$f_{(a,b)}(x)= [b-<a, x>](\text{mod } q) = b-\sum_{i=1}^{n} a[i].x[i]$

Le décryptage (a, b) est fait avec l'évaluation de la fonction linéaire f et la clé secrète sk (le résultat mod 2).

L'addition et la multiplication homomorphiques peuvent maintenant être décrites en termes de cette fonction f.

- ✓ L'addition : $f_{(a+a', b+b')}(x) = f_{(a,b)}(x)+f_{(a',b')}(x)$.

$= [[b-<a, sk>](mod\ q)]\ mod\ 2+ [[b'-<a', sk>](mod\ q)]mod\ 2$

L'ajout de deux cryptogrammes correspond à l'addition de deux fonctions linéaires, ce qui est encore une autre fonction linéaire.

- ✓ La multiplication :

$$f_{(a,b)}(x) \cdot f_{(a',b)}(x) = (b - \sum a[i]x[i]) \cdot (b' - \sum a'[i]x[i])$$
$$= h_0 + \sum h_i \cdot x[i] + \sum h_{i,j} \cdot x[i]x[j] ,$$

Cette opération traduit le degré du polynôme de n+1 à un polynôme de degré 2. C'est là la technique de re-linéarisation qui entre en jeu. La re-linéarisation est un moyen de réduire la taille du cryptogramme à n + 1. L'idée est de recalculer le message chiffré c en c' sous une autre clé sk', et donc la fonction devient :

$$h_0+\sum h_i[b_i-<a_i,sk'>] +\sum h_{ij}[b'_{ij}-<a_{ij},sk'>]$$

Cette technique permet à nouveau de chiffrer le message sous une nouvelle clé et faire la multiplication sans augmenter la taille du texte chiffré.

D. Chiffrement complètement homomorphique base sur RLWE (FHE from RLWE)2011:

Ce système est basé sur la technique Ring Learning With Error (RLWE) qui a été récemment introduite par Lyubashevsky, Peikert et Regev (Eurocrypt 2010) [34].
Toutes les constructions connues de cryptage entièrement homomorphique emploient une technique de «bootstrapping», qui impose la clé publique du système de croître de façon linéaire avec la profondeur maximale de circuits évalués. Ceci est un majeur inconvénient en ce qui concerne la facilité et l'efficacité d'utilisation.

La technique RLWE est équivalente à celle de LWE sauf qu'il utilise des anneaux différents (LWE utilise un anneau d'entiers R=Z et RLWE emploie un anneau de polynôme $R=Z[x]/x^d+1$).

E. Chiffrement complètement homomorphique par niveau (LFHE without Boostrapping):

C'est une nouvelle manière de construire des schémas de chiffrement complètement homomorphiques par niveau (LFHE) (capable d'évaluer les circuits arbitraires de taille polynomiale), sans la procédure de Gentry de bootstrap [35].

Spécifiquement, ce schéma se base sur les techniques FHE de LWE, ou FHE de RLWE, et il se caractérise par :

- ✓ L'addition d'une variante simplifiée de LWE et RLWE, dénommée GLWE « General Learning With Error »
- ✓ L'outil technique clé qu'ils utilisent pour la gestion du bruit est le «modulus switching » (changement de module)
- ✓ Une autre technique dite key switching permet le changement de clé, elle fait appel à deux procédure : BitBecompo et PawerOf2

Selon son nom «Leveled» FHE qu'il signifie : que cette technique s'exécute niveau par niveau, et opère la procédure de changement de clé à chaque niveau du circuit.

Changement de clé (Key Switching):

Cette technique est inspirée de celle de la technique de re-linéarisation de Brakarski, dont le principe est qu'à chaque niveau de multiplication on passe vers une nouvelle clé de chiffrement. Elle se compose de deux procédures:

- SwitchKeyGen : qui prend comme entrée les deux vecteurs clés secrètes, les dimensions respectives de ces vecteurs, et le module q, et en sortie des informations auxiliaires $\mathcal{T}s_1 \rightarrow s_2$ qui permet le changement (Switching).
- SwitchKey : qui prend l'information auxiliaire (résultat de la première procédure) et le message chiffré c1 sous la clé s1et en sortie un nouveau message chiffré c2 qui crypte le même message sous la clé secrète s2.

Les deux procédures précédentes utilisent certains sous-programmes, décrits comme suit :

- ✓ BitDecomp(x) : décompose x en sa représentation binaire.
- ✓ Powersof2(x, q) : externalise $(2^0.x, 2^1.x,\ldots,2^n.x)$.

Pour les deux vecteurs c et s de même taille, nous avons :

<BitDecomp(c,q), Powersof2(s,q)>= <c,s>mod q.

Changement du module (Modulos Switching) :

L'essence du Modulos Switching qui est une amélioration de la technique réduction de la dimension (LWE) [33]. Dans le changement de module de q à p, si p<q, l'erreur sera réduite, le module sera réduit proportionnellement, et donc le rapport taille de l'erreur / taille de module ne diminuera pas, cependant ce rapport détermine le nombre d'opérations homomorphiques possibles.

Considérons la nouvelle approche suivante. On choisit une échelle de modules décroissante $\{q_i \approx q/x_i\}$ pour $i<k$. Après, il faudrait, multiplier les deux textes chiffrés mod-q. La taille du bruit devient x^2. La réduction du module se fait par la formule suivante $q_1=q/x$. Le niveau de bruit du nouveau message chiffré va de x^2 vers x. Si en appliquant une autre multiplication sous le module q_1 la taille revient à x^2, maintenant en changent le module a $q_2= q_1/x$, puis le bruit redescend à x, ainsi de suite.

Cette nouvelle approche augmente le nombre de niveaux multiplicatifs (sans bootstrapping) qui est évalué par un facteur exponentiel. Cette amélioration exponentielle est suffisante pour atteindre LFHE sans bootstrapping.

Pour tout polynôme L, on peut évaluer les circuits de profondeur L $\{q \approx x^L\}$. Cette méthode permet d'évaluer efficacement un circuit arithmétique arbitraire de taille polynomiale (maintenir la quantité du bruit à un niveau constant) sans avoir recours à bootstrapping.

LFHE consiste à une amélioration exponentielle par rapport aux précédentes approches.

Optimisation :

1. **Boostrapping :**

   Bootstrapping n'est pas strictement nécessaire pour atteindre LFHE. Cependant, dans certains contextes, il peut avoir certains avantages. Parfois l'évaluation de circuit de profondeur L, avec L grand, la procédure de génération des clés est longue, et dès les premiers niveaux du circuit, la taille du texte chiffré devient grande. Pour pallier à ce problème, la procédure de Boostraping a été prise en charge.

2. **Batching :**

   Batching permet d'empaqueter plusieurs textes clairs dans un même texte chiffré de sorte que la fonction peut être évaluée homomorphiquement sur plusieurs entrées avec la même efficacité que d'être évaluée homomorphiquement sur une seule entrée. Un cas particulier est que lors du déchiffrement plusieurs texte chiffrés peuvent être lancé simultanément et de manière très efficace.

Synthèse des méthodes suivant le tableau au dessous:

Les tentatives du chiffrement FHE	**Critiques :**
A	Contrainte de performance: impact des opérations effectuées sur le bruit. Solution: application d'une procédure « boostrap » Non applicable en pratique: vu le temps de traitement et la taille du chiffré et des clés de chiffrement
B	Repose sur la précédente approche, génération des clés à partir du principe le plus grand diviseur en commun. Non applicable en pratique: taille importante des paramètres de sécurités
C	Résolution d'un système d'équation linéaire. Ré linéarisation pour obtenir SWHE réduire la taille du chiffré. Réduction dimension pour passer au FHE réduire le degré du polynôme.
D	RLWE et LWE deux problèmes identiques sauf qu'ils utilisent des anneaux différents.
E	Basé sur LWE et RLWE utilise : Key Switching à chaque niveau de multiplication, Modulos Switching pour maintenir le bruit, Possibilité d'effectuer K niveau de multiplication avant d'atteindre le plafond du bruit.

Tableau1 : Synthèse des méthodes de FHE.

Conclusion :

Le présent chapitre apporte les connaissances sur le chiffrement homomorphique qui est une nouvelle technologie dans le domaine de la cryptographie, ainsi que certaines approches de chiffrement complètement homomorphiques. Notre objectif est de présenter la technique « Leveled Fully Homomorphic Encryption » et ses méthodes de calcul que nous allons utiliser dans notre approche de la recherche d'information chiffrée (RIC) proposé dans la partie suivante.

Chapitre 3 :

La recherche d'informations sur les données chiffrées du Cloud Computing.

Introduction :

La recherche d'information sur les données chiffrées du cloud computing doit être simple et efficace en assurant la confidentialité des documents, contrairement à un système de recherche classique, les données sont toutes chiffrées aupréalable.

1. Le chiffrement des données (cryptographie) :

Le besoin de dissimuler les informations préoccupe l'homme depuis le début de la civilisation, avec les nouveaux moyens de communication est arrivée la nécessité d'assurer la confidentialité d'une partie de leur communications : l'origine de la cryptographie remonte sans doute aux origines de l'homme. Les premiers systèmes de cryptographie apparaissent vers 200 avant J.C. Les premiers outils mis en place permettait de rendre la lecture des informations difficile, et seule la complexité des mécanismes de cryptage était garantie de la confidentialité des messages. Mais l'avènement de l'Informatique et d'Internet la cryptographie prend tout son sens.

1.1. Définition :

La nécessité de cacher ou de casser une information rentre dans un vaste ensemble appelé cryptologie. La cryptologie est une science qui comporte deux branches : la cryptographie et la cryptanalyse.

- ✓ **La cryptographie :** la cryptographie traditionnelle est l'étude des méthodes permettant de transmettre des données de manière confidentielle. Afin de protéger un message, on lui applique une transformation qui le rend incompréhensible : c'est ce qu'on appelle le chiffrement, qui, à partir d'un texte en claire, donne un texte chiffré ou cryptogramme. Inversement, le déchiffrement est l'action qui permet de reconstruire le texte en claire à partir du texte chiffré. Dans la cryptographie moderne, les transformations en question sont des fonctions mathématiques, appelées algorithmes cryptographiques, qui dépendent d'un paramètre appelé clef [16].
- ✓ **La cryptanalyse :** à l'inverse, est l'étude des procédés cryptographiques dans le but de trouver des faiblesses et, en particulier, de pouvoir décrypter des textes chiffrés. Le décryptement est l'action consistant à retrouver le texte en claire sans connaitre la clef de chiffrement [17].

1.2. Vocabulaire de base :

La cryptographie utilise des concepts issus de nombreux domaines (Informatique, Mathématiques, Electronique), la Figure2.1 nous permet de définir le vocable de base [18]:

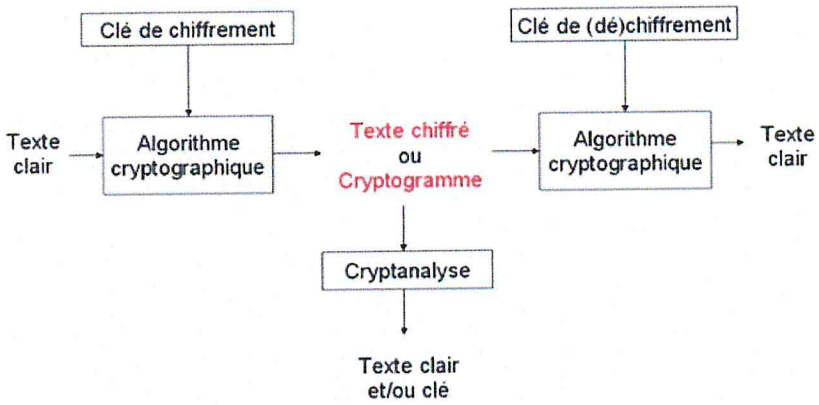

Figure3.1: Protocole de chiffrement[18].

- **Texte chiffré (crypté):** Appelé également cryptogramme, le texte chiffré est le résultat de l'application d'un chiffrement à un texte en clair.
- **Chiffrement (Cryptage):** Le chiffrement consiste à transformer une donnée (texte, message,...) afin de la rendre incompréhensible par une personne autre que celui qui a créé le message et celui qui en est le destinataire. La fonction permettant de retrouver le texte en clair à partir du texte chiffré porte le nom de déchiffrement.
- **Clef :** Il s'agit du paramètre impliqué et autorisant des opérations de chiffrement et/ou déchiffrement. Dans le cas d'un algorithme symétrique, la clef est identique lors des deux opérations. Dans le cas d'algorithmes asymétriques, elle diffère pour les deux opérations.
- **Cryptosystème (Protocole):** Il est défini comme l'ensemble des clés possibles (espace de clés), des texte en clairs et chiffrés possibles associés à un algorithme de chiffrement donné.
- **Signature :** La méthode réellement utilisée pour signer consiste à calculer une empreinte du message à signer et à ne chiffrer que cette empreinte. Le calcul d'une empreinte par application d'une fonction de hachage étant rapide et la quantité de données à chiffrer étant fortement réduite.

1.3. Les buts de la cryptographie [18]:

- ✓ Confidentialité: mécanisme pour transmettre des données de telle sorte que seul le destinataire autorisé puisse les lire.
- ✓ Intégrité : mécanisme pour s'assurer que les données reçues n'ont pas été modifiées durant la transmission.
- ✓ Authentification : mécanisme pour permettre d'identifier des personnes ou des entités et de certifier cette identité.
- ✓ Non-répudiation: mécanisme pour enregistrer un acte ou un engagement d'une personne ou d'une entité de telle sorte que celle-ci ne puisse pas nier avoir accompli cet acte ou pris cet engagement.

1.4. Les principaux concepts cryptographiques :

Cryptosystème à clé symétrique : La clé est identique, et doit être secrète

Au niveau de la génération des clés, elle est choisie aléatoirement dans l'espace des clés. Ces algorithmes sont basés sur des opérations de transposition et de substitution des bits du texte clair en fonction de la clé.

L'avantage principal de ce mode de chiffrement est sa rapidité, par contre le désavantage réside dans la distribution des clés : pour une meilleure sécurité, on préfèrera l'échange manuel. Malheureusement, pour de grands systèmes, le nombre de clés peut devenir conséquent. C'est pourquoi on utilisera souvent des échanges sécurisés pour transmettre les clés [18].

Cryptosystème à clés asymétrique :

Le principe de ce genre d'algorithme est qu'il s'agit d'une fonction unidirectionnelle à trappe. Une telle fonction à la particularité d'être facile à calculer dans un sens, mais difficile voire impossible dans le sens inverse. La seule manière de pouvoir réaliser le calcul inverse est de connaître une trappe. Une trappe pourrait par exemple être une faille dans le générateur de clés [18].

Au niveau des performances, le chiffrement par voie asymétrique est environ 1000 fois plus lent que le chiffrement symétrique. Cependant, à l'inverse du chiffrement symétrique où le nombre de clés est le problème majeur, en effet ici, chaque utilisateur possède une paire (SK, PK), et tous les transferts de message ont lieu avec ces clés. La distribution des

clés est grandement facilitée car l'échange de clés secrètes n'est plus nécessaire ainsi que la connaissance de PK ne permet pas de déduire SK. Chaque utilisateur conserve sa clé secrète sans jamais la divulguer. Seule la clé publique devra être distribuée.

Les algorithmes se basent sur des concepts mathématiques tels que l'exponentiation de grands nombres premiers (RSA), le problème des logarithmes discrets (ElGamal), ou encore le problème du sac à dos (Merkle-Hellman). L'algorithme de cryptographie asymétrique le plus connu est le RSA[18].

Fonction de Hachage :

C'est la troisième grande famille d'algorithmes utilisés en cryptographie,son principe est qu'un message clair de longueur quelconque doit être transformé en un message de longueur fixe inférieure à celle de départ. Le message réduit portera le nom de "Haché" ou de "Condensé". L'intérêt est d'utiliser ce condensé comme empreinte digitale du message original afin que ce dernier soit identifié de manière univoque. Deux caractéristiques (théoriques) importantes sont les suivantes [18]:

- Ce sont des fonctions unidirectionnelles : A partir de H(M) il est impossible de retrouver M.
- Ce sont des fonctions sans collisions : A partir de H(M) et M il est impossible de trouver M' $\neq$ M tel que H(M') =H(M).

Protocoles Cryptographiques :

Lorsque plusieurs utilisateurs sont impliquées dans un échange de messages sécurisés, des règles doivent déterminer l'ensemble des opérations cryptographiques à réaliser, leurs séquences, afin de sécuriser la communication, c'est ce que l'on appelle les protocoles cryptographiques. Lorsque l'on parle de "sécuriser un échange", on souhaite prêter attention aux trois services suivants : la confidentialité, l'intégrité et l'authentification.

Remarque : La distinction entre "services" (confidentialité, intégrité, etc.) et "mécanismes" (les moyens utilisés : chiffrement, signature, hachage, etc.) [18].

Dans le chapitre précédent, nous avons traités la recherche d'information ,et ce présent chapitre nous avons présentés, une briéve introduction sur la cryptographie, maintenant nous

sommes prêtes à entamer un concept qui est trés intéressant dans le cadre de ce projet a savoir : la recherche d'information sur les données chiffrées du cloud…

2. La Recherche d'informations cryptées du Cloud Computing:

Avec l'avènement du cloud computing, de nombreuses organisations et individus sont intéressés par l'externalisation de la gestion de données complexes aux cloud public pour l'épargne économiques et la facilité d'accès. Revers de la médaille, on laisse aux prestataires de ce service un accès total à nos informations ou, pire, ces dernières peuvent être interceptées par des personnes mal intentionnées. Pour résoudre ce problème, les données sensibles doivent être cryptées avant l'externalisation vers le cloud.

2.1. Définition :

De plus en plus les informations sensibles telles que des e-mails et des données financières sont entretenus par des professionnels dans des centres de données(data center ou cloud). Le fait que les propriétaires de données (data owner) et serveur cloud (cloud server) ne soient plus dans le même domaine de confiance peut mettre les données en clair en danger. Donc, avant l'externalisation sur le cloud, les données sensibles doivent être cryptées pour assurer leur confidentialité et la lutte contre l'accès non autorisé. Cependant, la recherche d'information traditionnnelle(non cryptées) par termes (Keywords) sur des textes en clair (plaintext) n'est pas fonctionnelle sur des données cryptées sur le Cloud.

Le système de la recherche d'information cryptés sur le cloud est constitué essentiellement des trois entités :

- Le propriétaire de données à externaliser ou data owner,
- Utilisateurs ou Users autorisé de faire la recherche d'information appelé aussi client,
- Serveurs Cloud qui héberge les documets .

La solution la plus simple est de télécharger toutes les données et les décrypter localement, cette solution n'est pas pratique et clairement irialisable, à cause de l'énorme quantité de données et le coût de la bande passante dans les systèmes à l'échelle de nuages.

Prenant en considération le grand nombre d'utilisateurs de données et les masses de données dans le Cloud, pour répondre à cette problématique, plusieurs approches de recherche sur des donnée cryptées ont été proposées[19].

D'abord, nous définissons le processus de recherche sur des données cryptées qui se base sur :

L'indexation des documents par le propriétaire de données (data owner), puis le chiffrement de l'index et de la collection de documents,

1. Partage des clés de chiffrement (secrètes et publiques) avec les utilisateurs autorisés,
2. Lors de la recherche un utilisateur génère un trapdoor (requête chiffrée) et l'envoie au serveur cloud,
3. Le cloud effectue la recherche sur l'index crypté et retourne à l'utilisateur une liste de documents cryptés (parfois un lien vers ces documents),
4. L'utilisateur décrypte cette liste et sélectionne les documents qu'ils l'intéressent.

Suivant ce processus chaque approche possède certaines caractéristiques et traitements sur les données, nous citons quelques méthodes qui ont été proposées par des chercheurs dans le domaine de la cryptographie.

2.2. Traveaux anterieurs:

A. Méthode traditionnelle de la recherche de données Cryptées (Traditional searchable encryption) :

Cette méthode peut assurer la recherche à travers un seul terme et récupérer des documents en utilisant la clé symétrique ou clé publique. Ils ne sont pas adaptés pour le système d'utilisation des données du cloud à grande échelle.

Certaines méthodes favorisant la recherche de terme booléenne (Boolean keyword search) sont conçues pour enrichir la flexibilité de la recherche tels que la recherche conjonctive et disjonctive. La recherche par termes conjonctive renvoie ces documents, y compris tous les termes , tandis que la recherche disjonctives renvoie chaque document qui contient même un seul terme. Elles ne sont pas adaptées pour fournir des résultats acceptables classés en fonction de la pertinence.

Pour récupérer des données efficaces dans la grande quantité de documents de stockage en nuage, il est nécessaire de présenter le résultat ordonné par pertinence [19].

B. Recherche d'informations cryptées à multi-mots-clés des données cryptées sur le cloud en deux étapes de classement (TSR):

TSR est basé sur les travaux de Xu et al [19] qui partent de l'hypothèse que l'*OPE* (*Order Preserving Encryption*) permet de préserver la distribution de la fréquence des termes pour avoir des résultats classés et précis.

OPE basé sur le protocole de la recherche d'information classée sur les données cryptées dans cloud, qui utilise la fréquence des termes chiffrée pour classer les documents et fournir des résultats pertinnents via la stratégie de classement TRS qui se résume en deux étapes:

- Classement des documents par catégories selon le nombre de termes de la requête, compris dans chaque document ce type de classement appelé aussi (*Coordinate Matching*).
- Classement des documents dans chaque catégorie à partir de la somme des fréquences des termes apparaissant dans chaque document (Summing scores in class).

Critiques :

Leurs inconvenient majeur est que la distribution et l'inter-distribution n'est pas caché et cela peu causer la fuite d'information sensible.

C. Recherche d'informations cryptées à multi-mots-clés des données cryptées sur le cloud en utilisant le schéma JL-Transform :

Li et al présentent une recherche multi-termes, son objectif est de réduire la taille de l'index, donc ils utilisent une méthode appelée la transformation de Johnson-Lindenstrauss (ou JL-Transform) [20].
Chaque document (requête) est représenté par un vecteur. Ils ont utilisés aussi la distance Euclidienne pour le calcule de similarité, puis retourner le top-k de documents qui sont les plus proches de la requête.

Le propriétaire de données Génére une clé K pour chiffrer l'index et une clé symétrique K' pour chiffrer la collection de documents, faire partager les clés avec les utilisateurs autorisés grâce à un protocole sécurisé de communication.

Il construit un index à partir de la collection de données, il tire toutes les termes ou chaque terme est pondéré avec la formule TF*IDF. L'index est sous forme d'une matrice ou les lignes représentent les documents et les colonnes représentent les termes du dictionnaire, et chaque élément de la matrice représente le poids du terme j dans le fichier i. La matrice (P) comporte beaucoup de zéros, parsque un fichier ne peut pas contenir tout les termes du dictionnaire. Pour couvrir les zéro et mettre l'index en sécurité ils ont utilsés cette démarche :

- Ajout d'un élément perturbateur (une matrice) à la matrice P (l'index en claire de taille n*m).
- Générer la matrice R de taille m*b, cette matrice R est calculée à partir d'un élément b, plus le b est grand, plus la taille de l'index I augmente et plus la précision augmente, le contraire est vrai. (projection JL-TRANSFORM).
- multiplier les deux matrice (R et P) pour avoir l'index crypté I de taille n*b (l'application de la JL-TRANSFORM).
- Publier (I, R).

Pour la construction de la requête cryptée (trapadoor) efficace, ils utilisent Optimized maximum query . Donc, il est plus intéressant d'améliorer le poids des termes importants de la requête.

Grace à la méthode JL-TRANSFORM il est possible de cacher la distribution des termes dans la requête, puisque le JL-Transform est une fonction à sens unique, les serveur ne peut pas connaître la fonction originale. Le serveur cloud ne peut pas connaître la relation entre deux Trapadoors (requêtes cryptées) car la même requête peut être chiffrée différement, et donc Trapadoor unlinkabilité est vérifié.

Le serveur est considéré comme honnête-mais-curieux, Il collecte des informations statistiques sur la requête, en déduisant que certains termes appartiennent à certains documents.

Critiques:

La JL-TRANSFORM peut protéger la distribution euclidienne mais pas le produit scalaire ni la similarité cosinus, donc ils ont été obligé d'utiliser la distance euclidienne même si elle est pas la plus efficace.

Le choix du paramètre b n'est pas mentionné pour la génération de la matrice R, donc un mauvais choix peut conduire à un résultat non-pertinant.

La séquence des résultats retournés n'est pas cachée du serveur cloud ce qui cause une fuite d'informaions.

D. Recherche d'informations cryptées sémantique sécurisée des données cryptées sur le cloud:

Sun et al [21] ont proposé une solution qui prent en charge la recherche sémantique classée sur les données chiffrées du cloud basée sur la cryptographie à clé symétrique.

La conception de cette approche propose une extension sémantique. Elle permet de retourner non seulement les documents exactement sélectionnés, mais aussi, les documents comprenant les termes qui sont sémantiquement liés aux termes de la requête.

Les métadonnées de documents sont construites pour chaque document de la collection. Une métadonnée représente tous les termes d'un document avec leurs poids associés.

Avec les métadonnées des documents, le serveur cloud construit l'index inversé, et un graphe sémantique appelé SRL pour les termes, donc réduit les tâches du propriétaire de données.

La relation sémantique entre deux termes est représentée par un score. A partir de ce score, un graphe sémantique serait construit (le SRL).

Cette approche exploite une fonction de hachage résistante aux collisions pour chiffrer les termes et un modèle OM-OPE pour chiffrer les scores.

Le modèle OPE cause une fuite d'information car le serveur peut identifier les termes de la requête à partir des analyses statistiques.

Sun et al, utilisent le modèle OM-OPE qui permet d'attribuer à un texte clair plusieurs textes chiffrés qui permet de cacher la distribution des termes.

L'utilisateur des données fournit une requête de recherche cryptée T_w de w termes au serveur cloud. D'abord le serveur cloud construit l'index inversé et le SRL en utilisant l'ensemble des métadonnées fournies par l'utilisateur de données. À la réception de la requête T_w, le serveur cloud est responsable de faire l'extension du terme de recherche sur le SRL. Ensuite, recherche l'index, et retourne les documents correspondants à l'utilisateur dans l'ordre.

Critique de l'approche :

Cette approche n'est pas fiable car c'est le serveur qui construit l'index. Mais, nous trouvons qu'elle est interéssante, car elle permet la recherche des termes qui sont liés sémantiquement mais, malheuresement, c'est le serveur qui s'occupe également de l'extension de la requête donc elle n'est pas sécurisée.

E. Recherche d'informations cryptées (Top- k) des données cryptées sur le cloud:

Yu et al [22] ont présenté une approche de recherche d'information sur le cloud computing en utilisant une nouvelle technologie dans la communauté de la cryptographie et de la recherche d'information: il s'agit du chiffrement homomorphique et la représentation vectorielle des documents et de des requêtes.

Le chiffrement homomorphique permet d'effectuer des traitements sur les données chiffrées et d'avoir des résultats chiffrés.

Après le déchiffrement de ce résultat on obtient les mêmes résultats que si on faisait la même opération sur les mêmes données en clair.

Le cloud computing est généralement considéré honnête-mais-curieux. Il exécute le protocole ou les traitements sur les données cloud correctement mais il analyse les données et les requêtes reçues de l'utilisateur pour apprendre des informations additionnelles par l'analyse statistique. Les informations qui pourra les atteindre sont la distribution[1] et inter-distribution [2] des termes. Donc, ces deux types de fuite d'informations doivent être cachées du cloud computing.

Pour eviter toutes fuite des informations sensibles, il faut que le score de similarité document/requête soit chiffré.

Cette approche introduit une notion :

Pour empêcher le cloud computing de déduire des informations, XU et al proposent qu' après avoir traité la requête sur le cloud. L'utilisateur s'occupera du tri des documents.

L'approche permet de représenter chaque document par un vecteur où chaque dimension de ce vecteur représente le poids du terme dans le document calculé par la

[1] C'est la distribution de la fréquence d'un terme dans chaque document de la collection.
[2] La distribution des scores des termes dans un document donné.

formule TF[3]*TDF[4], ainsi pour les requêtes sauf que, chaque dimension du vecteur requête représente si un terme apparait dans la requête ou pas (0 ou 1).

Dans le protocole proposé le propriétaire de données construit l'index puis chiffre l'index avec le chiffrement homomorphique. Lorsque le serveur cloud reçoit une requête multi-termes, il calcule le score de similarité entre la requête et chaque document puis envoie la liste des scores chiffrés à l'utilisateur. Ce dernier décrypte les scores et demande au serveur de récupérer les documents qui l'intéressent.

L'approche peut être divisée en deux phases : une phase d'initialisation, inclue Stup et indexBuild, et une phase de recherche qui comporte tapdoorGen, Score calculate et Rank.

- ✓ Phase d'initialisation :

  Le propriétaire de données fait appel à la fonction KeyGen pour générer la clé secrète SK et l'ensemble des clés publiques PK et communique SK aux utilisateurs autorisés.

  Il s'occupe aussi de l'extraction des termes de la collection formant un dictionnaire de termes puis,la construction de l'index I. Il le chiffre avec la fonction ENCRYPT. Les éléments d'un vecteur peuvent être cryptés avec les clés PK différentes. Le propriétaire de données crypte ainsi la collection C avec un autre protocole de cryptage ensuite il externalise la collection chiffrée C' et l'ndex I' sur le serveur cloud .

- ✓ Phase de recherche :

  L'utilisateur génére sa requête multi-terme. Il chiffre sa requête en fonction de TrapdoorGen pour obtenir T'. Le serveur cloud calcule la similarité entre requête document par le produit scalaire T'*I' et il retourne à l'utilisateur un vecteur N comme résultat de la recherche. Ce vecteur comporte tous les identifiants des documents avec leurs score de similarité calculé à partir de la requête utilisateur.

  L'utilisateur décrypte N et séléctionne les K premiers documents pertinants et envoie les identifiants des documents au serveur afin de lui renvoyer les K premier documents cryptés.

[3] Term frequency
[4] Invers terme frequency

Critiques:

L'utilisateur se charge lui-même de faire le tri de résultats retournés en réponse à sa requête.

L'homomorphique encryption permet de réduire le temps de chiffrement et de déchiffrement car elle a été adapté en une forme qui supporte que la multiplication et l'addition sur des entiers, ce qui permet de gagner en efficacité. Par conséquent les scores des termes d'un document doivent être des entiers plutôt que des réels donc il faut les arrondir, ce qui cause la diminution de la précision.

F. Recherche d'informations floue cryptées à multi-mots-clés des données cryptées sur le cloud:

Wangils et al [23] ont utilisé deux techniques pour la construction de cette approche:

- Bloom filter :

Un filtre bloom est un tableau de n bits qui sont mis à 0 au début. Il utilise les fonctions indépendantes de hachage LSH. Pour insérer un élément dans le tableau, ils utilisent les fonctions de hachage, donc elles retournent une position qui doit appartenir à la taille du tableau [1,n], et mettre la valeur à 1.

Pour vérifier si un élément appartient au filtre, il faut appliquer toutes les fonctions , si le résultat de cette position est à 0 c-à-d que l'élément n'appartient pas au filtre, sinon =1 si l'élément appartient ou il donne une fausse position.

- Locality-Sensitive-Mashing (LSH) :

Deux éléments dont leurs représentations consécutives sont proches l'un de l'autre et ont plus de chance d'avoir une même représentation par les fonctions LSH.

Cette approche élimine le besoin d'un pré-dictionnaire de termes, car le terme a une structure de données différente(bloom filter), qui permet la mise à jour des documents et donc l'index.

Contrairement aux solutions précédentes de la recherche multi-termes floue, le système fournir des résultats efficaces avec un « index » de taille constante quelque soit le nombre de termes dans le document, en exploitant des techniques puissantes comme le filtre de bloom, les fonctions de hachage LSH, la distance euclidienne, les bigrams, le produit scalaire.

Cette approche permet aux utilisateurs autorisés de formuler des requêtes multi-termes.

L'idée de base pour concevoir cette approche de recherche sécurisée des données chiffrées, ils ont pris en compte les éléments suivants :

- ✓ Une structure de données utilisée pour la représentation des indexes et des requêtes.
- ✓ L'algorithme de recherche efficace afin de calculer la correspondance entre requête – document.
- ✓ Les mécanismes de sécurité pour assurer la confidentialité de l'index et la requête.

Etapes principales de l'approche :

a. **Bigram vector representation of keyword :**

L'index est construit en récupérant les termes d'un document, ensuite chaque terme est représenté par un ensemble de bigram constitué de deux lettres contigües apparaissant dans le terme. ex : l'ensemble de bigrams du terme « network » est{ne, et,tw,wo,or,rk}

Chaque ensemble de bigrams d'un terme est représenté par un vecteur où chaque élément du vecteur représente l'un des bigrams possible. Une case du vecteur est à 0 si le bigram n'appartient pas au terme et égal à 1 sinon.

b. **Bloom filter representation of index query:**

Dans cette étape le bloom filter utilise les foncions de hachage ce qui permet la représentation des termes proches syntaxiquement. Ex « network » et « netword » ,la distance euclidienne entre les deux termes est inferieure à un certain seuil connu,

c. **Inner product and based matching algorithm :**

Ils ont représenté chaque terme par un ensemble de bigrams puis par un vecteur, tous les vecteurs seront enregistrés en fonction des fonctions de hachage dans un bloom filter (qui représente l'index)

Les requêtes seront construites de la même manière que l'index.

La mesure de similarité (score) entre requête-document est calculée en fonction du produit scalaire de vecteur document (bloom filter) et le vecteur requête(bloom filter)
D'après Wang et al, cette approche est résistible au Known background model.

- Key-word privacy (confidentialité de mot clé) : lors que le serveur cloud connait quelques informations sur l'ensemble des données, ces informations sont suffisantes et ont une forte probabilité d'identifier le mot-clé.
- Tapdoor unlikabiliby : tappdoor devrait être randomisé au lieu d'être déterministe, car la génération déterministe peut violer l'exigence « keyword-privacy »
- Access pattern (modele de base) : dans la recherche triée, le modèle d'accès est la séquence des résultats de recherche d'une requête W' noté par FW' constitue la liste des identifiants de tous les documents triés par pertinence.

MRSE-I cette méthode se base sur quatre algorithmes :

1. Steup : le propriétaire de données génère la clé secrète SK
2. BuildIndex (F,SK) : pour la construction index.
3. Trapdoor (W'):génération du trapdoor à partir du vecteur requête Q.
4. Query (I,W',K) : cette étape permet de retourner les K documents les plus proches par le serveur cloud en utilisant le produit scalaire de chaque document et la requête.

Dans, cette méthode le serveur cloud peut utiliser l'analyse statistique à grande échelle pour déduire des informations : fréquences de documents. La fuite de données est due à une valeur fixe d'une variable aléatoire dans le vecteur de données. Pour éliminer cette fuite, la méthode MRSE-II a été proposée.

Ils ont réglé cette fuite d'information par un ensemble de termes fictifs ajouté pour le vecteur de documents qui diminue l'attaque par l'analyse statistique.

Dans les vecteurs requête et le vecteur documents, la présence des termes est représentée par 1,mais un terme apparait dans un document à un seul endroit, comme il pourrait l'être dans plusieurs. Cette notion n'est pas représentée dans les modèles précèdents ce qui fait qu'ils ont introduit MRSE-I-TF et MRSE-II-TF qui permettent de mesurer les mots-clés pour leurs poids dans le document en appliquant la formule TF*IDF

Critiques :

Le majeur inconvénient de cette approche est que la séquence des résultats retournés à l'utilisateur (Access pattern) n'est pas cachée au serveur cloud qui peut causer une fuite des informations confidentielles.

Conclusion :

Dans ce chapitre nous avons présenté les principales notions et concepts de la recherche d'information, les systèmes de recherche d'information et des outils de recherche. Le but est de permettre aux utilisateurs de retrouver les documents dont le contenu répond à leur besoin en information, il s'agit donc de retourner l'ensemble des documents pertinents.

Nous avons présentés aussi le principe de base de la cryptographie qui permet de chiffrer les données de l'utilisateur pour préserver sa confidentialité. Nous avons exposé différents systémes de chiffrments. Nous avons étudié quelques approches de la recherche des données cryptées sur le cloud et leurs différentes failles de sécurité. Nous concluons que la recherche d'information, s'attache à définir des modèles et des systèmes afin de faciliter l'accès à un ensemble de documents et assurer la sécurité de ces derniers.

Partie II : Conception et Implémentation.

Chapitre 4 :

Conception d'une Approche RIC En Utilisant L'Homomorphic Encryption.

Introduction :

Dans cette partie, l'objectif principal est d'atteindre les besoins de la recherche d'informations cryptées RIC sur le cloud en conservant la confidentialité des données des utilisateurs, et leurs propriétés privées avec la nouvelle technologie utilisée par les communautés de la cryptographie et de la recherche d'information qui est le chiffrement Homomorphique « HE ». D'abord nous établissons un ensemble de contraintes et d'exigences, nous présentons notre modèle RIC, en utilisant la technique du chiffrement homomorphique afin de répondre à la problématique.

Le travail à réaliser est de proposer une solution efficace performante et sécurisée, pour une recherche d'information cryptée sur le cloud computing.

1. Position de la problématique :

L'avènement du cloud computing a offert aux entreprises l'opportunité d'externaliser leurs données en permettant un gain en termes de stockage et en gestion de données. Cependant l'externalisation de données sensibles induit à un problème de confidentialité substantiel. Afin de résoudre ces problèmes de sécurité cloud, des systèmes de recherche d'information cryptés sont apparus, qui utilisent le chiffrement de données et leur protection dans une certaine mesure.

Ces dernières années, plusieurs techniques de chiffrement ont été proposées pour la recherche sécurisée sur des données externalisées (étudier dans la partie précédente). Ces recherches se sont diversifiées avec des propositions d'amélioration du mode de recherches cryptées en essayant d'aboutir à une meilleure sécurité, flexibilité et performance.

Dans la majorité des approches proposées dans le domaine de la recherche d'informations cryptées RIC, le serveur cloud est considéré « honnête mais curieux » (honnest but curious) [19][20][21][22][23][24][25], c'est-à-dire le serveur suit honnêtement le protocole pour accomplir son rôle, cependant il peut effectuer des analyses statistiques sur les données stockées pour récolter des informations supplémentaires sur ses utilisateurs.

La problématique posée est de trouver une bonne représentation pour l'index/requête afin d'empêcher le cloud de faire des analyses statistiques sur les données sensibles, et de choisir une meilleure technique pour les chiffrer et les utiliser sur le serveur cloud sans avoir à les déchiffrer.

2. L'architecture du modèle proposé :

Notre processus de recherche d'informations cryptées se base sur trois différentes entités (comme montre la figure4.1): le propriétaire de données, l'utilisateur de données (client), et le serveur cloud.

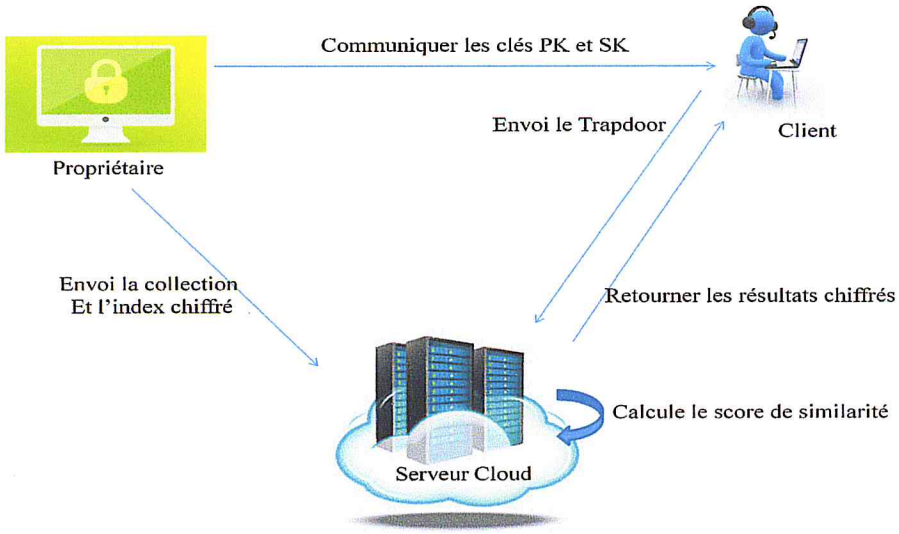

Figure 4.1: L'architecture de la recherche d'informations cryptées sur le cloud.

Nous présentons notre solution proposée sous forme d'un organigramme qui résume le fonctionnement du système RIC en général:

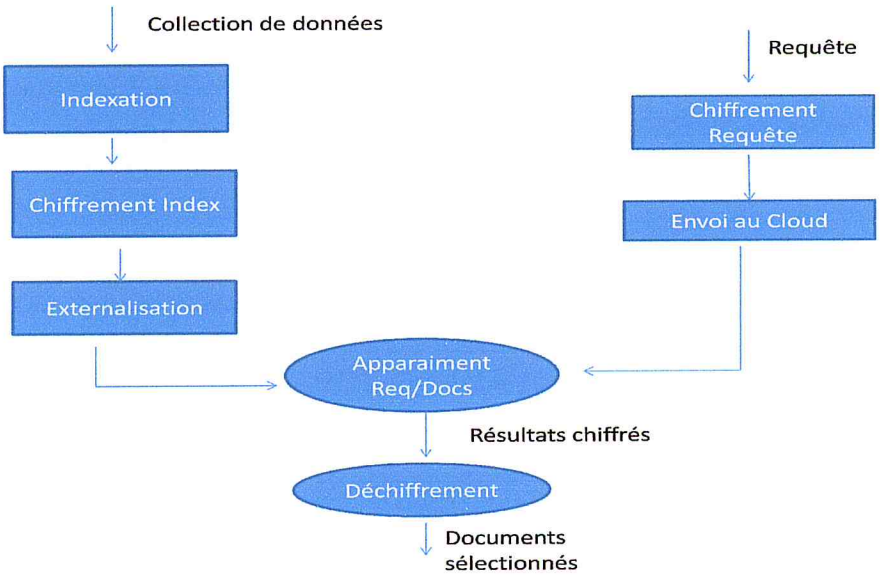

Figure4.2 : L'organigramme du système recherche d'informations chiffrées.

Avant que le propriétaire de données externalise la collection de documents (F), il la chiffre en utilisant une simple méthode chiffrement, et envoie cette collection cryptée (C) sur

le serveur. Ensuite, à partir de la collection F, il construit l'index (I), le chiffre avec le chiffrement homomorphique, et l'externalise sur le serveur. Après, un client autorisé peut construire une requête chiffrée et l'envoyer au serveur cloud. Le serveur cloud calcule la similarité entre l'index chiffré et la requête cryptée et retourne au client les scores chiffrés. Enfin le client choisit les documents qui l'intéressent.

Suivant notre protocole RIC, la solution proposée est au lieu d'utiliser une technique de chiffrement classique, nous avons employés l'Homomorphic Encryption spécifiquement Leveled Homomorphic Encryption pour le chiffrement de l'index et la requête.

3. Les contraintes :

Notre système de recherche d'informations chiffrées doit atteindre simultanément la sécurité et garantie la performance, pour cela nous établissons un ensemble de contraintes à respecter lors de la réalisation de cette approche:

1. Confidentialité des mots clés (Keywod Privacy) : le but est d'empêcher le serveur cloud (ou un hacker) de connaître les centres d'intérêt de l'utilisateur en déchiffrant sa requête, et de l'empêcher de faire le lien entre n'importe quel document de la collection et un ensemble de termes ;
2. Non-traçabilité du trapdoor (Trapdoor Unlinkability) : l'objectif de cette contrainte est d'empêcher le serveur cloud de déchiffrer les requêtes émises par les utilisateurs autorisés (les requêtes sont chiffrées avant leur envoi au serveur) et pour cela, la fonction de cryptage de requêtes doit être suffisamment aléatoire pour empêcher le serveur de faire le lien entre les différentes requêtes cryptées (trapadoors);
3. Protection des résultats de recherche (Search Pattern) : l'objectif de cette contrainte est d'empêcher le serveur cloud de connaître la séquence des résultats retournés à l'utilisateur afin de l'empêcher de faire des analyses statistiques.

4. Les exigences :

Les documents de la collection sont représentés par un index inverse. Tandis que la requête est représentée par un ensemble de termes associés à des poids (égale à 1).

La mesure de similarité du cloud est calculée par les opérations du chiffrement homomorphique à partir des données cryptées sur le cloud (requête/ index).

Nous avons proposé un système de recherche d'information chiffrée qui se base sur un chiffrement complètement homomorphique par niveau « LFHE » Leveled Fully Homomorphic Encryption[35].

5. La structure de données proposée :

Au début, on construit un dictionnaire de termes qui contient tous les termes de la collection, $W= \{w_1, w_2, w_3, \ldots, w_n\}$. La collection $F = \{d_1, d_2, d_3, \ldots, d_m\}$ serait cryptée et externalisée sur le serveur par le propriétaire de données. Chaque document de la collection est associé à un vecteur, où chaque dimension représente le poids du terme qui constitue le document.

La construction de l'index I, qui est un ensemble de termes extrait de la collection F, on associe à chaque terme un ensemble de document correspondant à ce terme où chaque document est représenté dans cette structure par un identifiant ID et deux valeurs TF et IDF qui permettent le calcule du poids de terme, à la fin nous obtiendrons un index inverse où ses entrées sont des termes.

Cette structure nous permet une recherche plus rapide et efficace car à la réception de la requête, le serveur va correspondre les termes de la requête aux termes de l'index, après il ne fait pas le calcul de similarité entre tous les documents de la collection et la requête (comme la recherche cryptée classique), mais seulement pour les documents qui satisfont la requête.

Ainsi pour la requête utilisateur, nous attribuons un ID aléatoire et un score pour chaque terme.

6. Description de l'approche proposée :

Notre schéma se base sur le chiffrement Homomorphique HE, qui permet d'effectuer des traitements sur les données chiffrées et d'avoir des résultats chiffrés. Après le déchiffrement de ces résultats on obtient les mêmes résultats qu'on fait la même opération sur les mêmes données en clair. Nous avons employé l'approche « Leveled Fully Homorphique Encryption » le chiffrement complètement homomorphique nivelé, présenté en 2012 par Zavika Brakerski, craig Gentry et Vinod Vaikuntanathan. (Bien décrite dans le précédent chapitre)

Notre choix s'est porté sur cette approche en raison de son efficacité, comparée aux autres approches de chiffrement complètement homomorphique (FHE) traitées dans le chapitre précédent, et ceci en terme de :

- ✓ Gain de temps de chiffrement et de déchiffrement.
- ✓ Optimisation de la taille du chiffré.
- ✓ Garantie de la confidentialité.

Le schéma proposé de l'approche de recherche d'information cryptée est composé de cinq algorithmes nommés : KeyGen, IndexBuild, TrapdoorGen, ScoreCalculate et Rank.

- ❖ KeyGen(λ) : le propriétaire des données génère une clé secrète SK, et la clé publique PK pour le chiffrement homomorphique. Cette procédure a comme entrée un paramètre de sécurité λ.
- ❖ IndexBuild(F, PK) : le propriétaire de données construit l'index de recherche à partir de la collection F, l'index I est sous format d'un fichier inverse où les entrée sont des termes et le contenu sont des identifiants de documents ID avec les valeurs TF et IDF correspondant aux termes, qui permet le calcule du poids par la formule TF x IDF.

Pour assurer la sécurité cloud en respectant les contraintes de l'approche proposée, une solution a été mise en œuvre pour rendre la tâche de l'attaquant difficile :

Chaque terme est associé à un ensemble de documents, pour empêcher le serveur de reconnaître les documents envoyés en réponse à une requête, la solution proposée est d'ajouter un ensemble de documents fictifs et mettre leurs poids à nul (vérifier la confidentialité des mots-clé ou Keyword Privacy), puis nous chiffrons l'index avec la clé publique de LFHE, et l'externalisations au cloud.

- ❖ TrapdoorGen(REQ,PK) : L'utilisateur génère un Trapdoor T à partir de sa requête multi-termes REQ, qui est représentée par un ensemble de termes exprimant les besoins de l'utilisateur. Pour améliorer le niveau de sécurité, nous ajoutons à cette dernière un ensemble de termes fictifs et attribuons les poids à tous les termes de la requête qui sont égaux à 1 sauf, le poids attribué aux termes fictifs est à 0 car il ne constitue pas la requête, les termes fictifs sont rajoutés juste pour empêcher le cloud de distinguer entre les requêtes utilisateur (vérifier la Non-traçabilité du Trapdoor ou Trapdoor inlikability), puis chiffrer les poids des termes de la requête avec la clé publique PK et envoyer T au cloud.

- ❖ Calcule du Score(T, I') : Le serveur cloud calcule le score de similarité entre le Trapdoor T et l'index I' (les documents retrouvés) (TF x IDF x Poid_du_terme) en fonction des opérations homomorphiques , puis il retourne les scores chiffrés à l'utilisateur (les résultats retournés inclut également des documents fictifs qui vérifier Access Pattern) .
- ❖ Rank(R, SK, k) : Après avoir reçu les scores de similarité, l'utilisateur décrypte les scores avec la clé secrète SK et trie les scores par ordre décroissant, puis envoie ce tri avec un paramètre k pour récupérer juste les k premiers documents pertinents.
- ❖ Serveur intermédiaire : Pour alléger les tâches de l'utilisateur, nous avons choisi à utiliser un serveur intermédiaire (tiers de confiance) entre l'utilisateur et le cloud qui s'occupe de : la formulation du Trapdoor (chiffrement de la requête) et le classement des résultats:

- ✓ Ajouter des termes fictifs à la requête,
- ✓ Attribuer les scores à chaque terme,
- ✓ Attribuer à chaque terme un ID identifiant qui est identique à ce attribué par le propriétaire à l'index,
- ✓ Chiffrer la requête obtenue avec la clé publique de l'LFHE
- ✓ Recevoir les résultats du cloud, déchiffrer les scores les triés et communiquer au client.

L'architecture de l'approche devient alors (figure 4.3) :

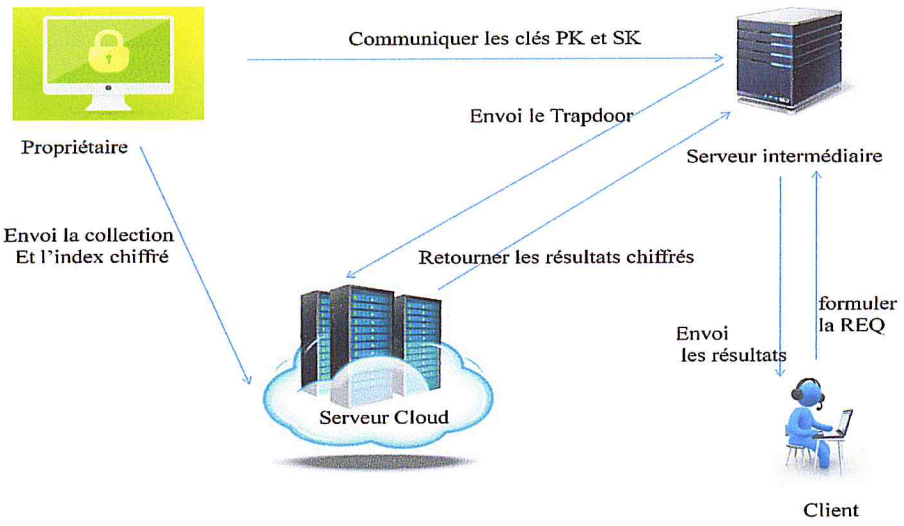

Figure4.3 :L'architecture RIC améliorée.

A ce niveau, Nous pouvons diviser notre approche en deux phases, phase d'initialisation et phase de recherche :

1. **Phase d'initialisation :**

   Le propriétaire de données exécute la fonction KeyGen pour générer la paire de clé, clé secrète SK et clé publique PK pour le chiffrement Leveled Homomorphic Encryption et envoyer la paire de clé au serveur intermédiaire.

   Ensuite, il extrait les termes à partir de la collection de documents et construit l'index inverse avec la fonction IndexBuild, puis calcule les valeurs TF et IDF associées à chaque terme.

   Le propriétaire attribue à chaque document un ID aléatoire, ainsi pour les termes (les ID attribué aux termes doivent être envoyé au serveur intermédiaire)

   Pour chaque ID de terme dans l'index inversé, il rajoute l'ensemble des documents fictifs avec un score nul. Après, il chiffre l'index avec la clé publique PK Homomorphique

   Enfin, le propriétaire chiffre la collection F par un autre algorithme de chiffrement classique (vu la complexité de la méthode choisi pour le chiffrement de l'index et le requête) et il externalise la collection chiffrée C et l'index chiffré *I'* dans le serveur Cloud.

2. **Phase de recherche :**

   L'utilisateur envoi sa requête (qui est un ensemble de termes) au serveur intermédiaire qui fait appel à la procédure TrapdoorGen pour la construction du Trapadoor. D'abord il ajoute un ensemble de termes fictifs à la requête, attribue les IDs et le score pour chaque terme de la requête, la chiffre afin d'obtenir un Trapdoor T et l'envoie au serveur cloud

   A la réception du Trapdoor T, le serveur Cloud fais appel à la procédure ScoreCalculate pour le calcul du score avec TF x IDF x des_Poids_terme entre l'index I' et T (multiplicition homomorphe) (Poids_terme associer à chaque terme de la requête) seulement pour les documents qui correspondent à la requête sans avoir à les déchiffrer. Il retourne les résultats au serveur intermédiaire, le serveur intermédiaire à son tour déchiffre le résultat, élimine les scores nuls, puis il fait le trie top-k des scores selon leur pertinence en faisant appel à la procédure Rank. Enfin il communique les résultats à l'utilisateur qui fait son choix correspondent à sa recherche.

7. Analyse de sécurité :

Nous avons supposés lors de la conception de notre approche, un ensemble de contraintes de sécurité exigées dans une méthode de recherche d'informations cryptées, nous expliquons comment l'approche répond à ces contraintes :

1. Confidentialité des mots-clés (keyWord privacy) : elle assure cette contrainte par : L'ajout d'un nombre fictif de documents dans chaque index inverse, l'attribution des ID_docs aléatoires pour chaque document, de même pour la requête pour chaque terme nous attribuons un ID aléatoire, et cela pour empêcher le cloud de faire le lien entre les termes de la requête et le contenu des documents de la collection.
2. Non traçabilité du Trapdoor (Trapdoor Unlikability): cette contrainte est respectée par : L'ajout des termes fictifs dans la requête, ce qui permet de générer pour une seule requête un Trapdoor différent à chaque recherche effectuées.
3. Protection des résultats de recherche (Search Pattern) : le résultat retourné est non seulement qu'il est chiffré, mais contient aussi un ensemble de documents fictifs, ainsi, le résultat retourné est différent pour une même recherche effectuée. Le serveur intermédiaire déchiffre les résultats, élimine les documents fictifs, puis, fait le classement, contrairement aux méthodes RIC étudiées où le Cloud qui fait le classement des résultats.

Et pour la protection du contenu (Protected Content) d'abord la collection de documents est chiffrée par un algorithme du chiffrement classique. Pour la construction de l'index, elle est faite par le propriétaire de données, et chiffrée par la méthode Homomorphyc Encryption, précisément l'approche Leveled FHE[35]. La requête est chiffrée par la même approche donc y a aucune chance d'avoir une fuite d'information.

Conclusion :

Dans ce chapitre nous avons présenté l'approche de recherche d'informations cryptées RIC proposée répond aux contraintes et atteint simultanément la sécurité et garantie la performance.

Chapitre 5 : Implémentation et Test

Introduction :

Nous allons présenter dans ce dernier chapitre l'environnement de travail et les différents outils pour le développement et la réalisation de notre contribution.

Nous allons mettre en œuvre l'approche conçue dans le précédent chapitre en utilisant le langage de programmation C++ qui dispose d'une certaine librairie appelé « HElib » qui est la dernière réalisation du chiffrement homomorphique.

1. Environnement et outils de développement :

1.1.Système d'exploitation :

Nous avons utilisé le système d'exploitation Ubuntu 32bits dans sa version16.04 pour notre implémentation sur une machine virtuelle.

1.2.Choix du langage :

Nous avons choisi d'utiliser le langage de programmation C++, qui décrit une robuste bibliothèque HElib que nous allons l'exploiter au cours de notre implémentation.

1.3.La librairie HElib :

1.3.1. Présentation de la librairie [37]:

Elle a été créée par Shai Halevi et Victor Shoup deux chercheurs du centre de recherche IBM Thomas J.Watson.

A la base, HElib est une implémentation de l'approche complètement Homomorphique de Gentry [27] se base sur les quatre algorithmes « KeyGen, Encrypt, Deccrypt, Eval », elle a été considérablement améliorée afin de perfectionner son fonctionnement.

HElib est structurée selon quatre couches :

- La couche de fond pour les modules de mis en œuvre des structures mathématiques,
- La seconde couche pour la représentation arithmétique des polynômes,
- La troisième concerne le crypto-système lui-même,
- La dernière couche qui est la couche supérieure relative aux interfaces pour l'utilisation des crypto-systèmes.

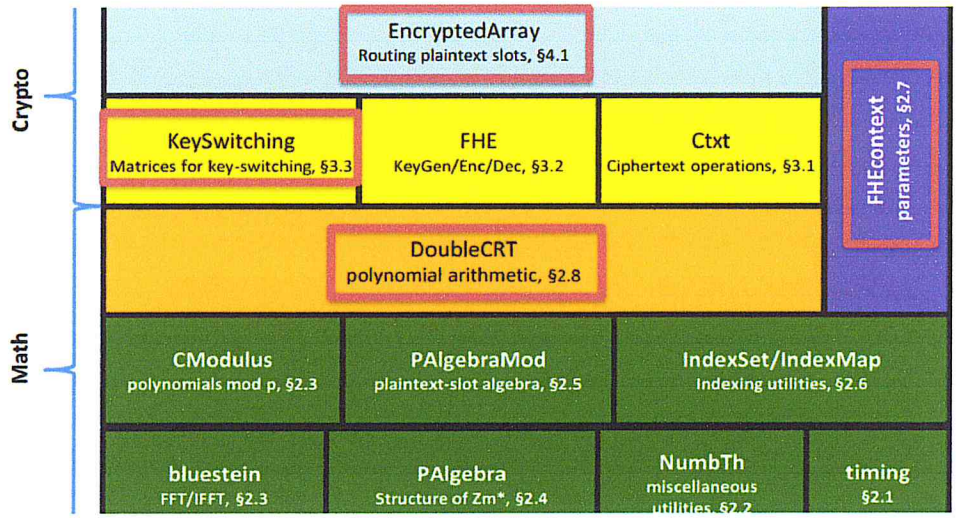

Figure5.1 : Schéma de la bibliothèque HElib[38]

La figure5.1 illustre la représentation de la bibliothèque ainsi que sa structure interne [38]:

Pour un fonctionnement correct de la librairie HElib sous C++, deux librairies mathématiques semblent nécessaires GMP et NTL. Elles permettent de travailler avec de "très grands nombres" (composés de centaines de milliers de chiffres, voire plus).

- GMP (GNU Multiple Precision Arithmetic Library): est une bibliothèque logicielle de calcul multi-précision sur des nombre entiers, rationnels et en virgule flottante. Les principaux domaines d'application de GMP sont la recherche et les applications en cryptographie, les logiciels applicatifs de sécurité pour Internet et les systèmes de calcul formel.
- NTL : est une bibliothèque logicielle de calcul arithmétique (théorie des nombres) sur des nombre entiers, multi-précision et en virgule flottante ainsi que les corps fini, les vecteurs, les matrices, les polynômes, et les bases de l'algèbre linaire.

1.3.2. Fonctionnement de la librairie:

HElib est une bibliothèque logicielle qui implémente l'Brakerski-Gentry-Vaikuntanathan (BGV) schéma de chiffrement homomorphique, ainsi que de nombreuses optimisations pour assurer la procédure d'évaluation homomorphe.

La librairie HElib comporte plusieurs classes comme illustre la précédente figure. Dans notre contexte nous exploitons seulement les classes de chiffrement et de déchiffrement définies ci-dessus.

✓ Configuration du Contexte :

En utilisant la classe *FHEcontext* qui nous permet, de stocker tous les paramètres nécessaires pour exécuter les autres opérations souhaitées. (Génération des clés, chiffrement, déchiffrement, opération homomorphe). L'initialisation des paramètres est comme suit :

```
long p=101 ;
long r=1 ;
long L=4 ;
long c=2 ;
long k=80 ;
long s=0 ;
long w=64 ;
long d=0 ;
long m= FindM(k, L, c, p, d, s, 0) ;
FHEcontext context (m, p, r) ;
```

P : est la base du texte en claire [default=2]

r : est le lifting [default=1]

L : est le nombre de niveaux

c : est le nombre de colonnes de la matrice key-switching[default=2]

k : est le paramètre de sécurité [default=80]

s : est le nombre minimum de slots [default=4] //emplacement

w : est le poids de Hamming de la clé secrète (utiliser dans la génération des clés dans l'approche BGV)

d : est le degré du champ d'extension [default=1]

(d==0 implique → factors[0]définie l'extension)

m : est un modulo spécifique

Nous avons également besoin de la valeur de G qui est un polynôme unitaire utilisé pour la construction de l'anneau de l'approche BVG.

```
buildModChain(context, L, c) ;
G= context.alMod.getFactorsOverZZ()[0] ;
```

✓ Génération des clés :

Cette procédure se fait après la configuration des paramètres, comme suit :

```
FHESecKey secretKey(context) ;
Const FHEPubKey& publicKey= secretKey ;
secretKey.GenSecKey(w) ;
```

✓ Chiffrement et Déchiffrement :

Suivant la génération des clés, les opérations de chiffrement et de déchiffrement peuvent être effectuées :

La librairie HElib, utilise une classe *EncryptedArray* pour faire ces opérations.

D'abord il faut faire appel à cette classe par l'instruction: #include <EncryptArray.h>
En suite, créer un vecteur de chiffrement comme suit :

```
EncryptedArray ea(context, G)
```

Puis faire le chiffrement sur le texte que nous voulons, notre texte est conservé sous un vecteur v :

```
Ctxt  c (publicKey) ;

ea.encrypt(c, publicKey, v) ;
```

Le déchiffrement se fait avec la clé secrète déjà générée : Le résultat du déchiffrement est sauvegardé dans le vecteur *res.*

```
ea.decrypt(c, secretKey, res) ;
```

✓ Opération Homomorphe :

Le texte chiffré est stocké dans une instance *Ctxt,* avec les opérateurs d'addition « += » et de multiplication « *= », nous effectuons des opérations sur ce texte chiffré afin de faire des calculs.

2. Implémentation :

Notre implémentation est basée sur l'utilisation de la bibliothèque HElib, afin d'aboutir au système de la recherche chiffrée proposé dans le précédent chapitre.

Ces étapes sont nécessaires pour atteindre notre objectif :

1. La génération et la sauvegarde des clés, ainsi que le chiffrement de l'index.
2. La compression de l'index.
3. La formulation et le chiffrement de la requête.
4. La recherche permet de faire correspondre la requête chiffrée avec l'index chiffré.
5. La décompression de l'index (seulement les documents qui correspond à la requête).
6. Le déchiffrement des résultats ainsi que leur classement.

2.1.La génération, la sauvegarde des clés, et le chiffrement de l'index :

Dans ce qui précède, nous avons proposé qu'un index est présenté sous forme d'un fichier inverse, dans notre implémentation nous allons le représenté sous format XML "Extensible Markup Language" (langage de balisage extensible).

Le choix s'est porté sur ce format, grâce à sa représentation adapter à la structure de données (notre cas l'index), facile à exploiter ou à produire, ainsi qu'il peu décrire tout domaine de données.

Le prototype de l'index est structuré de la manière suivante :

```
<?xml version = "1.0" encoding="UTF-8"?>
<Terme libelle= "terme" collection "/Collection">
<Document TF= "valeur"  IDF="valeur"> ID_Document</Document>
< /Terme>
```

Nous avons travaillé avec la bibliothèque RapidXML1.13 pour la manipulation des fichiers XML.

Cette étape comporte les procédures suivantes : init(), saveKeys(), readXML(), encryptXML(), et encryption().

La fonction GenSecKey() de la librairie HElib a pour la génération des clés du chiffrement, et la fonction init() permet l'initialisation des paramètres nécessaires, pour le chiffrement homomorphe.

saveKeys() permet de sauvegarder les clés dans les fichiers, SK et PK de la librairie HElib, pour l'utilisation du chiffrement de la requête et le déchiffrement des résultats.

readXML(), pour la lecture de l'index,

encryptXML(), pour le chiffrement de l'index (selon la conception)

Et voilà un aperçu pour la configuration de ces paramètres :

```
unsigned long m=4951, p=9001, k=80, L=8, r;
init(m, p, k, L);

void init(long p_m, long p_p, long p_k, long p_l)
{
     printf("Start initialisation ... \n");
long m=p_m, p=p_p, r=1;
long K = p_k;
long L = p_l;
long c = 3;
long d = 0;
long s = 0;
long chosen_m = 0;
long w = 64;

if (p_m==0)        m = FindM(K, L, c, p, d, s, chosen_m, true);
            //cout << m << endl;
context = new FHEcontext(m, p, r);
         //cout << context << endl;

buildModChain(*context, L, c);
secretKey = new FHESecKey(*context);
publicKey = secretKey;
secretKey->GenSecKey(w);
addSome1DMatrices(*secretKey);
cout << "Generated key " << endl;
G = context->alMod.getFactorsOverZZ()[0];
ea = new EncryptedArray(*context, G);}
```

Le chiffrement de l'index ce fait par le chiffrement des valeurs TF et IDF de chaque ID_document avec la fonction encryption() :

```
Ctxt encryption(long a){
//cout << "Start encryption ...\n" ;
vector<long> v1;
v1.push_back(a);
for (int i=1; i<ea->size(); i++)
v1.push_back(0);
Ctxt ct1(*publicKey);
ea->encrypt(ct1, *publicKey, v1);
//cout << "end encryption ...\n" ;
return ct1;
}
```

2.2.Chiffrement de la requête :

Pour le chiffrement de la requête, nous faisons appel à trois fonctions :

Loadkey() permet de récupérer les clés de chiffrement, readReq() pour la lecture de la requête (et l'ajout des termes fictifs), et la fonction encryption() pour le chiffrment.

```
Ctxt encryption(long a){
          //cout << "Start encryption ...\n" ;
vector<long> v1;
v1.push_back(a);
          //cout <<"a= " <<a<<endl;
for (int i=1; i<ea->size(); i++)
  v1.push_back(0);
  Ctxt ct1(*publicKey);
ea->encrypt(ct1, *publicKey, v1);
         //cout << "end encryption ...\n" ;
return ct1;
```

2.3.Compression Index :

Lors du chiffrement de l'index, sa taille devienne très importante du fait, nous avons proposé d'appliquer un algorithme de compression, sur l'index chiffré.

Parmi les algorithmes de compression, nous avons choisi un algorithme appelé « Zlib » sous C++.

Le format zlib a été conçu pour être compact et rapide pour une utilisation dans la mémoire et sur les canaux de communication. Le format gzip a été conçu pour la compression de fichier unique sur les systèmes de fichiers, a une tête plus grande que zlib pour conserver les informations de répertoire, et utilise une méthode de contrôle plus lent différent de zlib.

Zlib est une bibliothèque du C++, à partir de «`zlib.h` » nous utilisons des fonctions de compression `deflateInit()`, `deflate()`, et `deflateEnd()`.

2.4.Recherche :

Le programme recherche fait appel aux fonctions : loadKeys pour la récupération des clés, readEncryptXML pour lire l'index chiffré, et researtch().

La fonction researtch(), permet de chercher les termes de la requête dans l'index chiffré, si le termes existe nous récupérons les ID_documents et les scores compressés.

2.5.Décompression :

Après avoir récupérer les scores des ID_docs (ou l'index), nous appliquons la décompression sur ces derniers en exploitant la librairie Zlib.h

Notre index est un fichier XML, pour compresser une chaîne comme un document XML, faire exactement les deux instructions au-dessous:

```
ezbuffer buf;
ezcompress( buf, xml.GetDoc() );
```

En suite nous effectuons les calcules de similarités sur l'index décompressé (application opérations homomorphe TF x IDF x Poids_terme_requête).

2.6.Déchiffrement et classement :

Les deux procédures restantes pour accomplir notre système la fonction decryption() pour le déchiffrement et rank()pour le classement des résultats selon l'ordre de pertinence.

3. Test et performance :

Nous avons effectués des tests qui permettent l'évaluation des performances de la méthode du chiffrement homomorphe, en termes de temps et d'espace mémoire de données générées.

Nos résultats sont obtenu à partir d'un petit échantillon test, nous avons fixés un paramètre *dp* qui évalue la sensibilité des documents, il est défini sur une échelle de 0 à 9:

- ✓ Le niveau 0 : nous ne rajoutons aucun document fictif.
- ✓ Les autres niveaux (de 1 à 9) : le nombre de documents fictifs à ajouter calculé suivant cette formule : N/ (10-*dp*). Où N est le nombre total de document dans la collection, et *dp* est le niveau de sécurité souhaité sur cette échelle (0-9)
- ✓ Le niveau 9 les documents ajoutés égale aux nombre de documents dans notre collection.

Commençons par la génération la sauvegarde de clés (PK, SK), qui a duré 10 secondes, le temps de changement de clé effectué en 5 secondes. Ses opérations sont exécutées qu'une seule fois indépendamment de la taille de collection et de l'index.

Le temps de chiffrement de la requête fixé à dix termes est négligeable, vu que nous chiffrons que le poids du terme qui est zéro ou un.

Notre échantillon de l'index est représenté par 16 fichiers inverses, avec une taille de 150 Mégas Octet, son temps du chiffrement a duré 32 secondes.

Le temps de la recherche des termes dans l'index est égal à 28 secondes (selon nos données).

Le chiffrement de l'index, pour un fichier index en clair de taille 78,5 Kilos Octets, équivalent en chiffré est de 300 Mégas Octet d'où nous remarquons l'accroissement important de la taille de données.

Nous avons pris en considération cette taille importante, en appliquant un algorithme de compression sur l'index chiffré, afin de réduire sa taille et occupé moins d'espace mémoire.

Pour notre test les opérations de multiplication, (niveau deux de multiplication) et d'addition (une seule opération) exécutées en un temps de calcul insignifiant, vu le nombre d'opérations. Cependant, les opérations effectuées sur les données chiffrées permettent d'élever la taille du chiffré à un volume important.

Notre résultat nous permet de donner une idée concrète sur l'implémentation du chiffrement Homomorphe dans la pratique.

Notre solution apporte les performances suivantes :

- Notre système permet la recherche multi-termes comme la majorité des approches RIC.
- La structure de l'index proposée « index inverse » permet une recherche plus rapide contrairement à la structure classique (représentation vectoriel de l'index).
- L'ajout des documents est une considérable contrainte puisqu'il faut réindexer tous les documents et les chiffrer de nouveau, l'idée est de diviser le calcul du poids entre le propriétaire (calcule TF et IDF), et le Cloud (calcule le produit TF x IDF), qui conduit à une souplesse lors de la mise à jour de l'index.
- La recherche sémantique non prise en charge, contrairement à l'approche RSS [21]

- La recherche flou « fuzzy search » n'est pas prise en charge dans notre système contrairement à l'approche Privacy Preserving Multi-keyword Fuzzy search.[23]

Conclusion :

Dans ce présent chapitre nous avons pu utiliser la librairie HElib en exploitant ses différentes classes de Génération de clés, Chiffrement, Déchiffrement ainsi que les opérations homomorphe.

Lors des tests, nous avons obtenu un résultat très satisfaisant permettant de conclure que cette librairie répond bien aux besoins du chiffrement homomorphe, en particulier dans le domaine de recherche d'informations cryptées.

Conclusion Générale :

La cryptographie, dont on attend assure la confidentialité et l'authenticité des communications, subit actuellement un changement significatif, qui se traduit par des changements dans les paradigmes. Jusqu'à présent, la construction des primitives cryptographiques était considérée de point à point, entre deux entités, qui souhaitent échanger des données de façon sécurisée.

Cette cryptographie classique est très mal adaptée aux nouveaux usages, liés en particulier au développement du cloud. Le cloud permet de stocker des quantités de données extrêmement importantes, dans des fermes de serveurs sur lesquelles les usagers n'ont en réalité aucune garantie de sécurité. Plusieurs services de cloud computing permettent de télécharger des données en toute sécurité privées.

Cette sécurité peut empêcher un étranger de voir vos données alors qu'elles sont envoyées au cloud. Dans de nombreux cas, une fois que les données sont externalisées, le service de cloud peut décrypter les données. Les services de gestion de données cloud ne mettent en œuvre aucun protocole cryptographique adapté.

La cryptographie asymétrique est apparue lorsque Rivest, Shamir et Adelman publient leur crypto-système RSA. Ce crypto-système révolutionnaire dans le sens où il permet à tout le monde d'envoyer des messages chiffrés à une personne sans que personne hormis le destinataire ne puisse le déchiffrer. Ce crypto-système a donné naissance à une nouvelle ère, à savoir le chiffrement homomrphique.

Le chiffrement homomorphe est considéré comme l'un des éléments les plus prometteurs pour assurer la sécurité du Nuage. Malheureusement, la majorité des schémas existants entrainent une expansion de chiffré (la taille du chiffré par rapport à la taille du message initial) et des algorithmes de traitement lents. Des études approfondis, des contributions et des solutions ont été proposées afin de se débarrasser de ce fameux problème du chiffrement homomorphe et d'améliorer l'efficacité de ces crypto-systèmes.

Reste le chiffrement homomorphe le plus récent de la cryptographie moderne, son implémentation dans la pratique sera possible d'ici quelque année, ce qui révolutionnera le monde du cloud.

Cette étude à présenté cette modeste contribution, au domaine émergent de la recherche d'informations cryptée, consiste à étudier et appliquer le chiffrement Homomorphe en énumérant leur avantages et inconvénients.

Nos résultats permettent de donner une idée de la possibilité d'une utilisation concrète du chiffrement homomorphique dans le domaine de RIC. Des perspectives que nous pouvons apporter est de déployer notre solution posée dans un environnement cloud, afin de toucher mieux aux résultats (selon les données utiliser).

[34] Zavika Brakarski et Vinod Vainkuntanathan , Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages (2011)

[35] Zvika Brakerski, Craig Gentry et Vinod Vaikuntanathan , (Leveled) Fully Homomorphic Encryption without Bootstrapping(2012)

[36] Pierre Gérard, Introduction à UML 2 Modélisation Orientée Objet de Systèmes Logiciels, Université de Paris 13, DUT informatique-S2D

[37] La libraries HElib: http://shaih.github.io/HElib/ consulté le : 06/06/2016

[38] Ring---Learning With Errors & HElib Wenjie Lu, University of Tsukuba : riku@mdl.cs.tsukuba.ac.jp