Exploring the Scope of Machine Learning using Homomorphic Encryption in IoT/Cloud

Yulliwas Ameur

HESAM UNIVERSITÉ

le cnam

École Doctorale Sciences des Métiers de l'Ingénieur

Centre d'Études et de Recherche en Informatique et Communications

THÈSE DE DOCTORAT

présentée par : **Yulliwas AMEUR**

soutenue le : **18 Décembre 2023**

pour obtenir le grade de : **Docteur d'HESAM Université**

préparée au : **Conservatoire national des arts et métiers**

Discipline : **Section CNU 27**

Spécialité : **Informatique**

Exploring the Scope of Machine Learning using Homomorphic Encryption in IoT/Cloud

THÈSE dirigée par :
Mme Samia BOUZEFRANE Professeure, Cnam, France

et co-encadrée par :
M.Vincent Audigier Maître de Conférences, Cnam, France

Jury		
M. Pierre Paradinas	Professeur, Cnam,Paris École	Président
M. Joaquin Garcia-Alfaro	Professeur, SAMOVAR, Telecom SudParis, Institut Polytechnique de Paris	Rapporteur
M. Tim Hall	Docteur, NIST, Maryland, États-Unis	Rapporteur
M. Pascal Paillier	Expert Cryptographie, Zama	Examinateur

THÈSE

Je dédie cette thèse à mes parents, mes frères, ma femme, et mes enfants pour leur amour, leur soutien et leur encouragement constants tout au long de mon parcours académique.

I yimawlan-iw, i watmaten-iw, tameṭṭut-iw akked d warraw-iw.

Remerciements

Je tiens à exprimer ma profonde gratitude à ma directrice de thèse, Mme Samia Bouzefrane, pour sa confiance et son soutien inestimables. Sa guidance a été plus qu'un pilier académique, elle a été une source d'inspiration constante, éclairant mon chemin dans les moments de doute. Son engagement sans faille envers mon développement personnel et professionnel a transformé ce rêve d'enfance en réalité. Ses conseils avisés, sa patience inébranlable et son encouragement constant m'ont permis de naviguer avec assurance et détermination dans les méandres complexes de la recherche. Sa capacité à percevoir et à valoriser mes efforts, même les plus modestes, m'a donné la force de persévérer dans les périodes les plus exigeantes de ce parcours. Mme Bouzefrane n'est pas seulement une mentor académique ; elle est une véritable mentor de vie, dont l'influence perdurera bien au-delà de cette thèse.

Je suis également reconnaissant à mon co-encadrant, M. Vincent Audigier, dont les remarques pertinentes et l'accompagnement rigoureux ont grandement contribué à l'élaboration de ce travail. Sa perspicacité et son expertise ont été cruciales pour affiner ma pensée et ma compréhension des enjeux de notre domaine.

Je tiens à adresser un grand remerciement à M. Soumya Banerjee, Chercheur Adjoint. Sa collaboration, ses connaissances approfondies et son soutien ont été d'une valeur inestimable pour ma recherche. Son engagement et sa passion pour la science ont grandement influencé mon travail et ma vision académique.

Un merci particulier à M. Stefano Secci, le chef de l'équipe "Réseaux et Objets Connectés (ROC)" du Centre d'études et de recherche en informatique et communications (CEDRIC). Travailler au sein de son équipe a été une expérience enrichissante et formatrice. Sa vision et son leadership ont été des sources d'inspiration et de motivation tout au long de mon parcours de recherche.

Je souhaite également exprimer ma gratitude envers mes collègues de bureau au CNAM. Leur

compagnie, leur soutien et leurs discussions stimulantes ont grandement enrichi mon quotidien et contribué à mon épanouissement professionnel et personnel. Leur esprit de camaraderie a été un atout précieux tout au long de mon parcours de thèse.

Mes remerciements les plus chaleureux et respectueux s'adressent à mes rapporteurs, M. Joaquin Garcia-Alfaro et M. Tim Hall. Leurs analyses approfondies et leurs perspectives critiques sont un véritable trésor pour ce travail. Leurs commentaires perspicaces et leur regard expert sur ma recherche enrichissent mon travail et me guident vers une réflexion plus profonde et une compréhension plus nuancée de mon sujet. Leur capacité à cerner les aspects les plus subtils de ma thèse est une source d'inspiration et un défi stimulant qui élève incontestablement la qualité de mon travail.

Je tiens à exprimer ma gratitude particulière à M. Pierre Paradinas, qui a accepté de présider mon jury. Son expertise et son approche équilibrée sont des atouts précieux pour ma soutenance à venir. Sa présence et son leadership apporteront une dimension de rigueur et de sérieux, tout en créant un environnement propice à une discussion approfondie et constructive.

Un remerciement particulier également à M. Pascal Paillier, qui participera en tant qu'examinateur. Sa renommée en cryptographie et son expertise technique sont des éléments essentiels qui enrichissent ma recherche. Son rôle en tant qu'examinateur est non seulement un honneur mais aussi une opportunité exceptionnelle d'obtenir des retours d'un expert de son calibre.

Cette thèse est le fruit d'un travail collectif et je suis infiniment reconnaissant envers tous ceux qui ont contribué, de près ou de loin, à sa réalisation. Chaque rencontre, chaque discussion, et chaque mot d'encouragement ont été des pierres ajoutées à l'édifice de ce projet.

Je tiens également à exprimer ma gratitude envers tous les stagiaires que j'ai eu le privilège d'encadrer au cours de cette recherche, en particulier Rezak Aziz. Leur dévouement, leurs idées novatrices et leur contribution ont grandement enrichi l'environnement de travail et ont été des éléments clés dans le succès de ce projet.

Je suis profondément reconnaissant envers mes anciens professeurs du département de Mathématiques de l'Université Paris 8, en particulier pour le cours d'histoire de la cryptologie enseigné par Philippe Guillot, qui a été une source d'inspiration et un pilier dans mon parcours académique.

Enfin, je souhaite dédier ce travail à ma famille et à mes amis, dont le soutien inconditionnel a été le socle sur lequel j'ai pu construire et réaliser mes aspirations.

Chapitre 1

Résumé en français

1.1 Contexte de la thèse

Selon statista [1], 30,9 milliards de dispositifs IdO seront utilisés par les entreprises et l'industrie automobile d'ici à la fin de 2025. Cependant, ces appareils IoT ne disposent pas de ressources suffisantes pour traiter les données collectées par leurs capteurs, ce qui les rend vulnérables et susceptibles d'être attaqués. Pour éviter de traiter les données au sein des objets connectés, la tendance est à l'externalisation des données collectées vers le cloud, qui dispose à la fois d'une puissante capacité de stockage et de traitement des données. Cependant, les données externalisées peuvent être sensibles et les utilisateurs peuvent perdre leur confidentialité concernant le contenu des données, tandis que les fournisseurs de services en nuage peuvent accéder à ces données et potentiellement les utiliser pour leurs propres activités. Pour éviter cette situation et préserver la confidentialité des données dans le centre de données en nuage, une solution possible consiste à utiliser le chiffrement entièrement homomorphe (FHE), qui garantit à la fois la confidentialité et l'efficacité du traitement. Dans de nombreux environnements intelligents tels que les villes intelligentes, la santé intelligente, l'agriculture intelligente, l'industrie 4.0, etc. où d'énormes quantités de données sont générées, il est nécessaire d'appliquer des techniques d'apprentissage automatique (ML) et de contribuer ainsi à la prise de décision dans l'environnement intelligent. En effet, le défi dans ce contexte est d'adapter les approches d'apprentissage automatique pour qu'elles puissent être appliquées à des données chiffrées, afin que les décisions prises sur la base des données chiffrées puissent être traduites en données claires.

Nous examinons ces deux cas d'utilisation ci-dessous pour mieux illustrer le problème abordé dans cette thèse :

Cas d'usage 1 : Alice est diabétique et doit surveiller régulièrement son taux de glycémie à l'aide d'un lecteur de glycémie connecté à son smartphone. Elle a opté pour une application mobile de surveillance de la glycémie qui stocke ses données personnelles et de santé sur un serveur cloud tiers à des fins d'analyse.

Chaque fois qu'Alice mesure sa glycémie à l'aide de son lecteur, les données sont envoyées à l'application mobile, qui les stocke sur un serveur en nuage tiers à des fins d'analyse. Les données de santé d'Alice sont alors exposées à des risques potentiels tels que la violation de données, la surveillance non autorisée, l'usurpation d'identité, la fraude à l'assurance et même la discrimination en matière d'emploi ou d'accès aux soins de santé. La principale raison de l'externalisation des données vers le cloud est la mémoire et les ressources informatiques limitées du smartphone d'Alice, qui rendent difficile le traitement et le stockage de grandes quantités de données au niveau local. Cependant, cela pose des problèmes de sécurité et de confidentialité pour les données de santé sensibles d'Alice. Une solution possible à ce problème est l'utilisation du chiffrement entièrement homomorphe (FHE) pour protéger la confidentialité des données d'Alice tout en permettant une analyse et une prise de décision efficaces.

Cas d'usage 2 : Deux sociétés pharmaceutiques ont récemment fait des découvertes prometteuses dans la recherche d'un nouveau vaccin contre une maladie mortelle. Or, les deux entreprises disposent de données exclusives et complémentaires qui pourraient contribuer à accélérer le processus de recherche et de développement du vaccin.

Cependant, les entreprises sont réticentes à partager leurs données car elles ne se font pas confiance et craignent que leurs secrets commerciaux ne soient dévoilés. C'est là que le chiffrement homomorphique entre en jeu. Les entreprises peuvent désormais crypter leurs données respectives et les envoyer à un tiers de confiance qui effectue l'agrégation et le regroupement sans jamais avoir à décrypter les données ou à révéler leurs secrets commerciaux.

Grâce au chiffrement homomorphique, les entreprises peuvent collaborer en toute sécurité et sans risque en partageant leurs données pour la recherche et le développement de vaccins. Les deux entreprises peuvent bénéficier de l'expertise et des connaissances de l'autre tout en préservant la confidentialité de leurs données commerciales. Dans la section suivante, nous présentons un exemple introductif qui a motivé le travail de cette thèse.

1.2 Objectifs et contributions de la thèse

Les travaux réalisés dans le cadre de cette thèse ont donné lieu à plusieurs publications et ont étudié l'application du chiffrement homomorphe dans différents contextes d'apprentissage automatique. Le premier travail [2] se concentre sur l'utilisation du chiffrement homomorphique dans un environnement multi-cloud pour améliorer la sécurité et la confidentialité des données. Pour préserver la vie privée et la confidentialité lorsque les données sont externalisées, une plateforme multi-cloud est proposée, qui intègre des clouds publics, privés et gérés avec une interface utilisateur unique. Les données hébergées dans le nuage sont distribuées à différents centres de données dans un environnement multicloud, en tenant compte de la fiabilité du nuage et de la sensibilité des données. Le chiffrement homomorphique est utilisé à cet effet, une méthode de chiffrement qui permet de traiter et de manipuler les données tout en restant chiffré, de sorte que les utilisateurs ou les tiers peuvent traiter les données chiffrées sans révéler leur contenu.

La deuxième partie de cette thèse étudie l'application du chiffrement homomorphique à l'algorithme k-nearest neighbors (k-NN). L'étude [3] présente une mise en œuvre pratique de l'algorithme k-NN utilisant le chiffrement homomorphique et démontre la faisabilité de cette approche sur un grand nombre d'ensembles de données. Ce travail aborde le problème de la vulnérabilité liée à l'externalisation des données dans le nuage. La solution proposée utilise un schéma de chiffrement homomorphe (appelé TFHE) pour l'algorithme k-NN, qui permet un traitement des données chiffrées de bout en bout tout en préservant la vie privée. Contrairement aux techniques existantes, cette solution ne nécessite aucune interaction intermédiaire entre le serveur et le client pendant la tâche de classification. L'algorithme a été évalué sur des ensembles de données réels importants et pertinents et a prouvé son efficacité en s'adaptant bien à différents paramètres sur des données simulées.

Dans la troisième partie de cette thèse, l'application du chiffrement homomorphique à l'algorithme de regroupement k-means est étudiée. Comme pour l'étude k-NN, le travail proposé présente une implémentation pratique de l'algorithme k-means en utilisant le cryptage homomorphique et évalue sa performance sur différents ensembles de données.

Enfin, nous présentons une autre contribution [4] qui utilise la combinaison du chiffrement homomorphique avec des techniques de confidentialité différentielle (DP) pour renforcer davantage la confidentialité des modèles d'apprentissage automatique. Les travaux proposés suggèrent une nouvelle

approche qui combine le chiffrement homomorphique et la protection différentielle afin d'obtenir de meilleures garanties de confidentialité pour les modèles d'apprentissage automatique.

En résumé, la recherche présentée dans cette thèse complète la littérature croissante sur la convergence du chiffrement homomorphique et de l'apprentissage automatique, et fournit des implémentations pratiques et des évaluations du chiffrement homomorphique dans divers scénarios d'apprentissage automatique.

1.3 Contribution à la Gestion des Problèmes de Sécurité via le Chiffrement Homomorphique dans un Environnement Multi-Cloud

Cette contribution examine l'externalisation des données vers des plateformes cloud, soulignant les défis de sécurité et de confidentialité. Elle propose l'utilisation d'une plateforme multi-cloud pour renforcer la confidentialité et la disponibilité des données. La contribution présente une plateforme multi-cloud intégrant des clouds publics, privés et gérés avec une interface utilisateur unique. Cette approche optimise la distribution des données en fonction de la fiabilité du cloud et de la sensibilité des données. Elle souligne également les limites des algorithmes de chiffrement actuels, notamment leur coût élevé en ressources. La contribution met en lumière le chiffrement homomorphique comme moyen de traiter des données cryptées sans révéler leur contenu. Cette méthode est particulièrement adaptée aux environnements où la confiance envers les fournisseurs de cloud est limitée.

1.3.1 Application et Défis Identifiés

L'étude se focalise sur l'application du chiffrement homomorphique dans la sécurisation des environnements multi-cloud. Elle identifie des défis existants et des opportunités d'amélioration, en mettant un accent particulier sur les applications dans le domaine des dossiers médicaux électroniques.

1.3.2 Contribution Principale

La principale contribution est une nouvelle architecture pour les dossiers médicaux électroniques dans un environnement multi-cloud, utilisant la bibliothèque open-source "OpenFHE" pour le chiffrement homomorphique. Cette architecture vise à sécuriser le partage de données dans un contexte multi-cloud, avec une démonstration de faisabilité pour des opérations homomorphiques simples, et des

perspectives pour des opérations plus complexes comme l'application d'algorithmes d'apprentissage automatique.

1.3.3 Conclusion

Cette contribution souligne l'importance de relever les défis de la sécurité dans les applications cloud, particulièrement dans le secteur de la santé. Elle présente une architecture innovante qui assure la confidentialité dans un environnement multi-cloud, en utilisant un algorithme de chiffrement homomorphique multi-clés.

1.4 Classificateur k-NN Sécurisé et Non-Interactif Utilisant le Chiffrement Homomorphique Symétrique

Cette contribution aborde la problématique de la vulnérabilité liée à l'externalisation des données sur le cloud dans le contexte du "Machine learning as a service" (MLaaS). Elle propose une solution pour l'algorithme des k plus proches voisins (k-NN) utilisant un schéma de chiffrement homomorphique (TFHE), permettant le traitement des données entièrement chiffrées tout en préservant la confidentialité.

1.4.1 Contexte et Défis

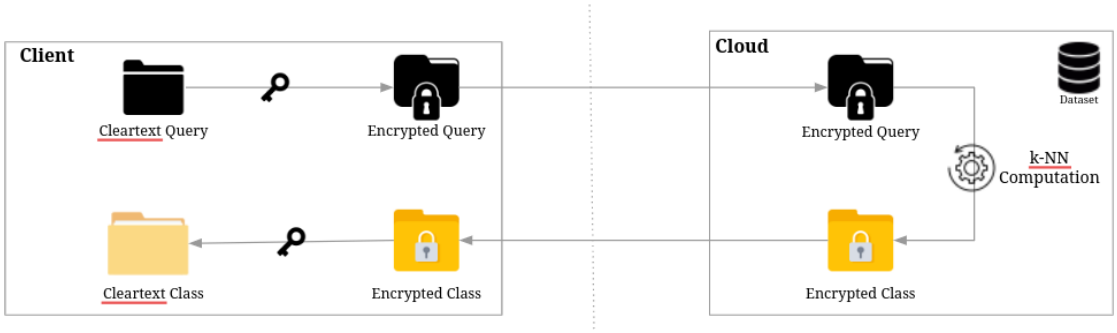
Les techniques cryptographiques courantes pour préserver la confidentialité dans l'apprentissage automatique incluent le partage secret, le calcul multipartie et le chiffrement homomorphique (HE). Cette contribution se concentre sur l'utilisation du chiffrement homomorphique pour résoudre les défis liés à l'application de l'algorithme k-NN dans un environnement chiffré.

1.4.2 Contribution Principale

La principale innovation est la méthodologie permettant d'appliquer le k-NN sur des données chiffrées en utilisant le chiffrement homomorphique complet, évitant toute interaction entre les entités. Cette solution garantit qu'aucune fuite d'information ne se produit pendant le processus, et elle supporte la classification multi-labels. Les étapes clés incluent le calcul de distance, le tri des distances, la sélection des k voisins les plus proches, et le vote majoritaire, tous réalisés sur des données chiffrées. Le

système utilise une architecture client-serveur, où le client (le demandeur) envoie une requête chiffrée au serveur (propriétaire des données) pour exécution. Le propriétaire des données exécute l'algorithme k-NN chiffré et renvoie le résultat au client pour déchiffrement.

1.4.3 Le modèle du système



-0.5cm

FIGURE 1.1 – Le modèle du système : le client est le demandeur et le serveur est le propriétaire des données : le propriétaire des données reçoit la requête de manière cryptée, exécute un algorithme k-NN crypté puis envoie le résultat au demandeur pour décryptage.

-0.5cm

Notre système utilise l'architecture client-serveur (voir figure 1). Le client est le demandeur et le serveur est le propriétaire des données.

propriétaire des données : il possède les données et peut effectuer des calculs lourds. Par exemple, il reçoit la requête de manière cryptée, exécute un algorithme k-NN crypté, puis envoie le résultat au demandeur pour qu'il le décrypte. demandeur : génère les clés, crypte la requête qui contient ses données et l'envoie au propriétaire des données pour qu'il effectue des calculs avant de décrypter le résultat. Le demandeur peut être un ordinateur ordinaire ou tout appareil IoT qui collecte des données.

1.4.4 Défis chiffrés k-NN

Afin de proposer une version chiffrée du k-NN, nous devons remplacer les opérations de défi utilisées dans le k-NN standard par des opérations équivalentes dans des domaines chiffrés. Comme nous l'avons vu précédemment, le k-NN est composé de trois parties : le calcul de la distance, le tri de la distance, la sélection des k plus proches voisins et le vote majoritaire. Cette sous-section présente les opérations équivalentes telles qu'elles sont intégrées dans notre solution.

1.4.4.1 Calcul de la distance

Le calcul de la distance euclidienne entre les entrées de l'ensemble de données x_i et la requête q est nécessaire pour trouver les k plus proches voisins de la requête. Nous pouvons utiliser la formule standard de la distance (1).

$$d^2(x_i, q) = \sum_{j=0}^{p} x_{ij}^2 + \sum_{j=0}^{p} q_j^2 - 2 * \sum_{j=0}^{p} x_{ij} q_j \tag{1.1}$$

Ce qui est important dans notre cas, c'est la différence entre deux distances pour les comparer. Nous obtenons donc la formule (2) suivante :

$$d^2(x_i, q) - d^2(x_{i'}, q) = \sum_{j=0}^{p} (x_{ij}^2 - x_{i'j}^2) - 2 * \sum_{j=0}^{p} (x_{i'j} - x_{ij}) q_j \tag{1.2}$$

Étant donné que l'ensemble de données est un texte clair, nous pouvons facilement calculer la formule (2) à l'aide du schéma TFHE. Toutefois, nous devons l'adapter. Avec TFHE, la différence entre les distances doit être comprise entre $[-\frac{1}{2}, \frac{1}{2}]$. Une autre contrainte est que la multiplication est effectuée entre un entier en clair et un texte chiffré. Deux valeurs de remise à l'échelle sont nécessaires pour résoudre ces contraintes. La première est v. Elle est utilisée pour obtenir les valeurs des différences entre $[-\frac{1}{2}, \frac{1}{2}]$. Soit p le second. Il indique la précision des différences. Chaque attribut de l'ensemble de données et de la requête est rééchelonné à l'aide de v. p est utilisé pour calculer le produit $(x_{i'j} - x_{ij})q_j$.

1.4.4.2 Tri

Le tri des distances calculées est une étape cruciale du k-NN. L'algorithme standard de tri, tel que le tri par bulles, peut être utilisé en considérant des données cryptées. Cependant, ces algorithmes prennent beaucoup de temps dans un monde crypté car le pire cas est calculé à chaque fois. Les auteurs [5] proposent deux méthodes pour trier un tableau de valeurs. La méthode du tri direct est utilisée dans [6]. Elle est basée sur une matrice de comparaison appelée matrice delta :

$$\begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,n} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,n} \end{pmatrix}$$

avec

$$m_{i,j} = \bar{sign}(X_i - X_j) = \begin{cases} 1 & \text{if } X_i < X_j \\ 0 & \text{else.} \end{cases}$$

En additionnant les colonnes de cette matrice, nous aurons un indice de tri des distances.

1.4.4.3 Vote majoritaire

Le vote majoritaire peut poser un problème car l'opération nécessite une comparaison pour détecter la classe. Pour déterminer la classe prédite, nous devons connaître les classes de k voisins les plus proches. Cette étape est difficile à deux égards : tout d'abord, nous devons éviter les fuites d'informations, contrairement aux solutions proposées dans la littérature. Deuxièmement, le vote majoritaire nécessite une comparaison afin de prédire la classe.

À notre connaissance, aucune solution dans la littérature n'a étudié ce point de manière cryptée sans fuite d'information. Par conséquent, dans la sous-section suivante, nous démontrons une solution pour traiter k-NN avec le vote majoritaire de manière cryptée tout en prenant en charge une classification multi-label.

1.4.4.4 Notre algorithme k-NN proposé

L'algorithme que nous proposons, appelé "HE-kNN-V", se compose de trois étapes : la construction de la matrice delta, la sélection des k plus proches voisins et le vote majoritaire. Les deux premières étapes sont similaires à la solution de [6] même si nous adaptons les formules existantes afin d'éliminer les calculs inutiles. [6] utilise des polynômes pour définir les formules, alors que ce qui nous intéresse est un seul terme de ces polynômes pour éliminer les calculs non nécessaires. Le vote majoritaire est notre valeur ajoutée et est spécifique à notre solution. Nous examinerons dans cette sous-section la conception de notre solution, y compris chaque élément constitutif.

-sectionConstruction de la matrice delta Pour construire la matrice delta, nous devons connaître le signe des différences entre les distances de tri. Puisque nous avons défini une méthode pour calculer les différences dans la dernière sous-section, le signe peut facilement être obtenu en utilisant la fonction de signe bootstrapping standard dans TFHE. Cependant, la fonction bootstrapping standard renvoie +1 si la phase est supérieure à 0 et -1 si la phase est inférieure à 0. Par conséquent, puisque nous devons avoir 0 ou 1 dans la matrice, nous devons adapter l'opération bootstrapping pour renvoyer $\frac{1}{2}$ et $-\frac{1}{2}$ puis, en ajoutant $\frac{1}{2}$ au résultat, nous obtiendrons 0 ou 1.

Même si la construction de cette matrice prend du temps, elle est hautement parallélisable.

1.4.4.5 Sélection des k plus proches voisins

Pour sélectionner les k–voisins les plus proches, nous utilisons l'opération de notation proposée par Zuber [6]. En utilisant la matrice delta, le principe est le suivant :

> m valeurs dans chaque colonne avec m le nombre d'opérations possibles sans bootstrapping. §'il reste des valeurs à additionner : effectuer une opération de bootstrapping à l'aide de la fonction de bootstrapping de signe modifiée (voir l'algorithme 1 dans [6]) et passer à l'étape 1.Sinon, exécuter la fonction d'amorçage de signe modifiée et renvoyer le dernier signe renvoyé par cette opération.

Enfin, nous obtenons un vecteur crypté où la position i est égale au chiffre 1 si l'individu ayant l'indice i fait partie des k plus proches voisins, au chiffre 0 sinon. Nous appelons ce vecteur le "masque" (voir la figure 2 pour plus de clarté).

-sectionVote majoritaire Le vote majoritaire est la valeur ajoutée la plus importante de notre travail. Nous proposons d'effectuer le vote majoritaire sans aucune fuite d'information, contrairement aux travaux existants comme celui de [6] dans lequel la valeur de la majorité est effectuée en texte clair ou en utilisant d'autres solutions alternatives proposées dans la littérature. èrement, nous illustrons le problème avec la méthode de [6]. Nous considérons le scénario dans lequel le demandeur effectue les calculs. Le vote majoritaire est effectué en texte clair, mais nous devons décrypter le vecteur d'index du plus proche voisin. Le propriétaire des données effectue le décryptage. Une fuite d'informations importante se produit si le propriétaire des données connaît le vecteur d'index. Il connaîtra alors la classification de la requête et, en effectuant une triangulation, il pourra approximer la requête. En outre, la solution sera interactive. Si nous considérons le scénario dans lequel le propriétaire des données effectue le calcul, le décryptage du vecteur est effectué par le demandeur. Cependant, pour effectuer la classification, le demandeur doit connaître les étiquettes de l'ensemble de données, ce qui constitue également une fuite d'informations critique. En outre, le demandeur connaîtra la taille de l'ensemble de données et le paramètre k des plus proches voisins considérés. Ces informations sont considérées comme des informations internes du modèle utilisé et doivent être protégées.

Dans notre solution, le vote majoritaire est effectué par le propriétaire des données de manière cryptée. Tout d'abord, le propriétaire des données code les étiquettes à l'aide d'un codage à chaud. Le masque et la matrice des étiquettes étant sous une forme chaude, il est facile d'effectuer une opération

ET entre le masque et chaque colonne des étiquettes, comme le montre la figure 2. Nous obtenons une matrice A (pour affectation) avec A_{ij} égal à 1 si l'individu i fait partie des k plus proches voisins et que sa classe est j. En utilisant cette matrice, il est possible de faire la somme des colonnes et d'obtenir la probabilité de chaque classe. On peut alors ne renvoyer que la classe et garantir l'absence de fuite d'information et d'interactivité.

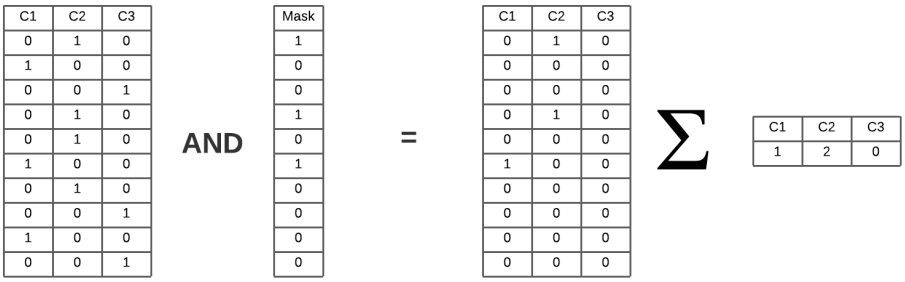


FIGURE 1.2 – Illustration du vote majoritaire à l'aide du masque

1.4.5 Évaluation de la Performance

La méthode a été évaluée en termes de temps d'exécution, de précision et de consommation de bande passante sur de grands ensembles de données réels. Les résultats montrent que la solution est efficace et évolutive, offrant une précision comparable à celle obtenue avec des données en clair.

1.4.6 Conclusion et Perspectives

HE-kNN-V propose une méthode pour exécuter k-NN sur des données chiffrées, incluant un vote majoritaire pour l'attribution de classe. Cette solution aborde toutes les étapes de l'algorithme k-NN avec des données entièrement chiffrées, éliminant le besoin d'interactions intermédiaires entre le serveur et le client lors de l'exécution des tâches de classification. En perspective, l'accélération matérielle du schéma TFHE pourrait améliorer le temps de calcul de la solution proposée HE-kNN

1.5 Résumé de Contribution : Clustering k-means Sécurisé Utilisant le Chiffrement Homomorphique Complet sur le Torus (TFHE)

Cette contribution traite de l'application sécurisée du clustering k-means dans le cloud en utilisant le chiffrement homomorphique complet (FHE). Malgré les avantages du cloud pour le machine learning,

la confidentialité des données reste une préoccupation majeure, notamment dans les secteurs sensibles. Pour y remédier, cette recherche utilise la méthode TFHE (Fast Fully Homomorphic Encryption over the Torus) pour effectuer des opérations cryptographiques sur les données, tout en préservant la confidentialité. La contribution examine les techniques existantes d'HE appliquées au clustering k-means, y compris les défis liés à la réalisation de calculs complexes tels que la division et la comparaison dans un contexte cryptographique.

1.5.1 Contribution Principale

L'innovation principale réside dans la réduction de l'interaction entre le serveur cloud et l'utilisateur/client en utilisant TFHE. Cette approche permet d'exécuter toutes les étapes de l'algorithme k-means, y compris l'attribution des données aux clusters, de manière entièrement cryptée et non interactive. La méthode suggérée se distingue par sa capacité à effectuer des comparaisons chiffrées pour l'attribution des clusters sans déchiffrement intermédiaire.

1.5.2 Mise en Œuvre et Évaluation

La solution est mise en œuvre en C/C++ avec la bibliothèque TFHE et testée sur divers ensembles de données. Les performances sont évaluées en termes d'efficacité (comparaison avec k-means standard), de temps d'exécution et de sécurité. Les résultats montrent que l'algorithme proposé produit des clusters comparables à ceux du k-means standard, avec un niveau de sécurité élevé et des temps d'exécution pratiques pour des ensembles de données de taille modeste.

1.5.3 Conclusion et Perspectives

Cette contribution présente une méthode pratique et sécurisée pour l'exécution du clustering k-means dans le cloud, offrant une solution viable pour les applications nécessitant la confidentialité des données. Les perspectives incluent l'optimisation de la solution pour une exécution plus rapide, notamment en utilisant le parallélisme GPU, et l'exploration d'autres techniques de chiffrement homomorphique pour la sécurisation du k-means.

1.6 Contribution : Développement du Chiffrement Homomorphique Adaptatif en Explorant la Technique de Confidentialité Différentielle

Cette étude se penche sur les technologies de préservation de la confidentialité (PPTs) en réponse aux vulnérabilités inhérentes aux systèmes cloud. L'objectif est d'améliorer l'utilité tout en maintenant des normes strictes de confidentialité. Le chiffrement homomorphique et la confidentialité différentielle (DP) sont combinés pour renforcer les applications d'apprentissage automatique contre des adversaires non conventionnels.

1.6.1 Mise en Œuvre

Le travail expérimente avec l'ensemble de données Breast Cancer Wisconsin, traité pour préserver la confidentialité. Le chiffrement homomorphique, spécifiquement le schéma de Paillier, est utilisé pour assurer un transfert et des calculs sécurisés.

Code Côté Client

Des fonctions telles que `storeKeys()`, `getKeys()`, `serializeData()`, et `load_prediction()` sont utilisées pour gérer la génération, le chiffrement et le déchiffrement des données.

Code Côté Serveur

Le serveur exécute des calculs homomorphiques sur les données chiffrées et renvoie la prédiction chiffrée au client.

1.6.2 Analyse de Performance et de sensibilité

L'ajout de bruit gaussien impacte légèrement la précision, mais le modèle démontre une bonne performance, soulignant l'efficacité des techniques de préservation de la vie privée.

L'analyse de la sensibilité est cruciale pour réguler la quantité de bruit ajouté et assurer l'efficacité de la DP. Le modèle proposé vise à modéliser la sortie de la base de données avec une distribution de bruit appropriée.

1.6.3 Conclusion et Perspectives

Cette recherche propose une stratégie fondamentale pour combiner chiffrement homomorphique et DP. L'architecture client-serveur démontrée justifie la compatibilité des bibliothèques pour le chif-

frement homomorphique. Les recherches futures pourraient explorer le développement de mécanismes robustes minimisant le bruit tout en améliorant la confidentialité.

1.7 Conclusion générale

La présente thèse a entrepris une exploration complète des implications de l'utilisation du chiffrement homomorphique dans le domaine de l'apprentissage automatique en tant que service (MLaaS) et de la protection des données sensibles. L'expansion rapide du MLaaS a soulevé des préoccupations importantes concernant la confidentialité des données sensibles, ce qui nous a incités à étudier le chiffrement homomorphique en tant que solution viable. Des recherches approfondies ont été menées sur son application à divers aspects de l'apprentissage automatique, y compris son déploiement dans un environnement multi-cloud, son intégration dans l'algorithme k-nearest neighbors (k-NN) et son adaptation à l'algorithme de clustering k-means.

Les résultats de cette thèse ont démontré de manière concluante que l'utilisation du chiffrement homomorphique peut assurer efficacement la sécurité des données sensibles tout en permettant des opérations de traitement de données complexes. Notre étude a révélé que malgré les complexités associées à la mise en œuvre du chiffrement homomorphe, ses performances restent comparables à celles des méthodologies non chiffrées, présentant ainsi des opportunités convaincantes pour renforcer la protection des données sensibles dans le domaine de l'informatique en nuage.

Ces découvertes prometteuses ouvrent des voies stimulantes pour de futures recherches scientifiques. L'amélioration des composants matériels des schémas de cryptage homomorphique est une perspective qui mérite d'être envisagée pour renforcer les capacités de traitement. En outre, un examen approfondi de l'intégration du chiffrement homomorphe avec des méthodologies complémentaires de protection de la vie privée, telles que la protection différentielle de la vie privée, pourrait renforcer la sécurité des modèles d'apprentissage automatique dans les paradigmes sensibles.

En résumé, cette thèse représente une contribution substantielle à l'avancement des connaissances sur la sécurisation des données sensibles dans les environnements MLaaS. Les résultats servent à stimuler l'exploration en cours et la mise en œuvre plus large du chiffrement homomorphique, ouvrant ainsi de nouvelles possibilités pour sécuriser les données sensibles dans les applications d'apprentissage automatique et d'informatique en nuage.

De nombreux défis doivent être relevés pour appliquer l'apprentissage automatique préservant la vie privée dans des applications réelles. Bien que les normes, les plateformes et les mises en œuvre du chiffrement homomorphe décrites dans ce chapitre contribuent à l'avancement du chiffrement homomorphe dans l'apprentissage automatique (HEML), il reste des défis spécifiques à relever, notamment en ce qui concerne les frais généraux, les performances, l'interopérabilité, les goulets d'étranglement au démarrage, la détermination des signes et les cadres communs :

1. **Overhead** : Par rapport à son homologue non chiffré, HEML s'accompagne d'un surcoût important, ce qui le rend inadapté à de nombreuses applications. Toutefois, pour les modèles non HE, la phase d'apprentissage du ML implique un effort de calcul intensif. Même avec des techniques modernes, cela devient de plus en plus difficile avec HE. Une tendance récente consiste à contourner l'étape d'apprentissage en utilisant des modèles pré-entraînés afin de trouver un équilibre entre la complexité et la précision.
- **Parallélisation** : L'incorporation d'algorithmes bien établis et nouveaux est une approche permettant de faire face à la surcharge de calcul. Les ordinateurs à haute performance, les systèmes distribués et les ressources spécialisées peuvent tous être utilisés dans les modèles HEML. Les unités de traitement multicœur (GPU, FPGA, etc.) et les puces personnalisées (ASIC) fournissent des environnements HEML plus efficaces et plus conviviaux. Une autre approche pour améliorer l'efficacité globale consiste à regrouper et à paralléliser de nombreuses opérations d'amorçage.
- **Comparaison et fonction min/max** : De nouvelles méthodes sont nécessaires pour comparer les nombres cryptés par chiffrement homomorphique (HE). Actuellement, les fonctions de comparaison et min/max sont évaluées à l'aide de fonctions booléennes dans lesquelles les nombres d'entrée sont cryptés bit par bit. Cependant, les méthodes de cryptage bit à bit nécessitent des calculs relativement coûteux pour les opérations arithmétiques de base telles que l'addition et la multiplication.
- **PPML (Privacy Preserving Machine Learning) tools** : La conception d'une solution PPML performante et sécurisée sans une compréhension approfondie de l'apprentissage automatique constitue un défi pratique pour le déploiement de ces technologies. Les développeurs de PPML ont besoin d'une expertise à la fois en apprentissage automatique et en sécurité. Le PPML, qui utilise l'apprentissage automatique, n'a pas été largement accepté par la communauté de l'ap-

prentissage automatique en raison de la barrière d'entrée élevée de l'apprentissage automatique et du manque d'outils conviviaux.

— **Protocoles hybrides** : L'adoption de protocoles hybrides, qui combinent deux protocoles ou plus afin de tirer parti de leurs avantages et d'éviter leurs inconvénients, est une voie prometteuse pour l'amélioration des performances.

Abstract

Machine Learning as a Service (MLaaS) has accelerated the adoption of machine learning techniques in various fields. However, this trend has also raised serious concerns about the security and privacy of the sensitive data used in machine learning models. To address this challenge, we use homomorphic encryption. The aim of this thesis is to investigate the implementation of homomorphic encryption in different machine learning applications. The first part of the thesis focuses on the use of homomorphic encryption in a multi-cloud environment, where the encryption is applied to simple operations such as addition and multiplication. This thesis investigates the application of homomorphic encryption to the k-nearest neighbors (k-NN) algorithm. The study presents a practical implementation of the k-NN algorithm using homomorphic encryption and demonstrates the feasibility of this approach on a variety of datasets. The results show that the performance of the k-NN algorithm with homomorphic encryption is comparable to that of the unencrypted algorithm. Third, the paper investigates the application of homomorphic encryption to k-means clustering algorithm. Similar to the k-NN study, this paper presents a practical implementation of the k-means algorithm using homomorphic encryption and evaluates its performance using different datasets. Finally, the thesis explores the combination of homomorphic encryption with Differential Privacy (DP) techniques to further improve the confidentiality of machine learning models. The study proposes a novel approach that combines homomorphic encryption with DP to achieve better privacy guarantees for machine learning models. The research results presented in this paper contribute to the growing body of research at the intersection of homomorphic encryption and machine learning and provide practical implementations and evaluations of homomorphic encryption in various machine learning contexts.

Table des matières

Liste des tableaux

Table des figures

Première partie

General Introduction

Chapitre 2

General Introduction

2.1 Context of the thesis

According to statista [1], 30.9 billion IoT devices will be in use for companies and the automotive industry by the end of 2025. However, these IoT devices do not have sufficient resources to process the data collected by their sensors, which makes them vulnerable and susceptible to attacks. To avoid processing the data within the connected objects, the trend is to outsource the collected data to the cloud, which has both powerful data storage and processing capacity. However, the outsourced data can be sensitive and users may lose their privacy regarding the content of the data, while the cloud providers can access this data and potentially use it for their own activities. To avoid this situation and maintain data privacy in the cloud data center, one possible solution is to use fully homomorphic encryption (FHE), which guarantees both confidentiality and efficiency of processing. In many smart environments such as smart cities, smart health, smart agriculture, Industry 4.0, etc., where huge amounts of data are generated, it is necessary to apply machine learning (ML) techniques and thus contribute to decision-making in the smart environment. Indeed, the challenge in this context is to adapt machine learning approaches so that they can be applied to encrypted data, so that decisions made on the basis of the encrypted data can be translated into clear data.

We consider these two use cases below to better illustrate the problem addressed in this thesis :

Use case 1 : Alice is a diabetic and has to monitor her blood glucose level regularly with a blood glucose meter connected to her smartphone. She has opted for a mobile glucose monitoring application that stores her personal and health data on a third-party cloud server for analysis.

Whenever Alice measures her blood glucose level with her meter, the data is sent to the mobile application, which stores it on a third-party cloud server for analysis. Alice's health data is then exposed to potential risks such as data breaches, unauthorized surveillance, identity theft, insurance fraud and even discrimination in employment or access to healthcare.

The main reason for outsourcing the data to the cloud is the limited memory and computing resources on Alice's smartphone, which makes it difficult to process and store large amounts of data locally. However, this poses security and privacy risks for Alice's sensitive health data. One possible solution to this problem is the use of fully homomorphic encryption (FHE) to protect the confidentiality of Alice's data while still enabling efficient analysis and decision making.

Use case 2 : Two pharmaceutical companies have recently made promising discoveries in the search for a new vaccine against a deadly disease. However, both companies have exclusive and complementary data that could help speed up the research and development process for the vaccine.

However, the companies are reluctant to share their data because they do not trust each other and fear that their trade secrets will be exposed. This is where homomorphic encryption comes into play. Companies can now encrypt their respective data and send it to a trusted third party who performs the aggregation and clustering without ever having to decrypt the data or reveal their trade secrets.

Thank you to homomorphic encryption, companies can collaborate securely and without risk in sharing their data for vaccine research and development. Both companies can benefit from each other's expertise and knowledge while maintaining the confidentiality of their business data.

In the following section, we present an introductory example[7] that motivated the work of this thesis.

2.2 Objectives and Contributions of the thesis

The work carried out as part of this thesis has led to several publications and investigated the application of homomorphic encryption in different machine learning contexts. The first work [2] focuses on the use of homomorphic encryption in a multi-cloud environment to improve the security and privacy of data. To maintain privacy and confidentiality when data is outsourced, a multi-cloud platform is proposed that integrates public, private and managed clouds with a single user interface. The data hosted in the cloud is distributed to different data centers in a multi-cloud environment, taking into

account the reliability of the cloud and the sensitivity of the data. Homomorphic encryption is used for this, an encryption method that allows data to be processed and manipulated while remaining encrypted, so that users or third parties can process encrypted data without revealing its content.

The second part of this thesis investigates the application of homomorphic encryption to the k-nearest neighbors (k-NN) algorithm. The study [3] presents a practical implementation of the k-NN algorithm using homomorphic encryption and demonstrates the feasibility of this approach on a large number of datasets. This work addresses the vulnerability problem related to outsourcing data to the cloud. The proposed solution uses a homomorphic encryption scheme (called TFHE) [8] for the k-NN algorithm, which enables end-to-end encrypted data processing while preserving privacy. Unlike existing techniques, this solution does not require any intermediate interactions between the server and the client during the classification task. The algorithm has been evaluated on large and relevant real-world datasets and has proven its efficiency while scaling well over different parameters on simulated data.

In the third part of this thesis, the application of homomorphic encryption to the k-means clustering algorithm is investigated. Similar to the k-NN study, the proposed work presents a practical implementation of the k-means algorithm using homomorphic encryption and evaluates its performance on different datasets.

Finally, we present another contribution [4]that utilizes the combination of homomorphic encryption with differential privacy (DP) techniques to further strengthen the privacy of machine learning models. The proposed work suggests a novel approach that combines homomorphic encryption with DP to achieve better privacy guarantees for machine learning models.

In summary, the research presented in this thesis complements the growing literature on the convergence of homomorphic encryption and machine learning, and provides practical implementations and evaluations of homomorphic encryption in various machine learning scenarios.

Deuxième partie

Background and State of the Art

Chapitre 3

Background

3.1 Machine Learning Techniques

Machine learning is a branch of artificial intelligence (AI) that deals with the development of algorithms and statistical models that enable a computer system to perform certain tasks by learning from data rather than being explicitly programmed. Formally, machine learning can be defined as a collection of data analysis techniques that use algorithms to learn from training data and generalize the acquired knowledge to make predictions or decisions about new data. The goal of machine learning is to improve the performance of the system based on the data obtained over time and minimize the need for human intervention to adjust the algorithm's behavior. The theoretical foundations of machine learning come from various fields, including statistics, optimization theory, information theory and computer science, and there are numerous practical applications in computer vision, natural language processing, finance, healthcare and cybersecurity.

let X be the data set used for learning. We consider the following general machine learning tasks :

Classification : Let Y be the set of possible categories. A classification function $c : X \to Y$ maps each item $x \in X$ to a category $y \in Y$. Mathematically, classification can be formalized as finding a function c that minimizes the expected loss, $L(c(X), Y)$, where L is a loss function that measures the discrepancy between the predicted categories and the true categories.

Regression : Let Y be the set of possible real values. A regression function $r : X \to Y$ maps each item $x \in X$ to a real value $y \in Y$. Mathematically, regression can be formalized as finding a function r that minimizes the expected loss, $L(r(X), Y)$, where L is a loss function that measures the discrepancy

between the predicted values and the true values.

Ranking : Let Y be the set of possible rankings. A ranking function $f : X \to Y$ maps each item $x \in X$ to a ranking $y \in Y$. Mathematically, ranking can be formalized as finding a function f that minimizes the expected loss, $L(f(X), Y)$, where L is a loss function that measures the discrepancy between the predicted rankings and the true rankings.

Clustering : A clustering algorithm partitions the set X into K subsets (clusters) $C_1, C_2, ..., C_K$, such that each subset contains items that are similar to each other and dissimilar to items in other subsets. Mathematically, clustering can be formalized as finding a partition $C_1, C_2, ..., C_K$ that minimizes a certain objective function, such as the sum of squared distances between items and their cluster centers.

Reduction of Dimensionality : Let X be a high-dimensional vector space and Y be a lower-dimensional vector space. A dimensionality reduction algorithm maps each item $x \in X$ to a lower-dimensional representation $y \in Y$, while preserving some properties of the initial representation, such as distances between items. Mathematically, dimensionality reduction can be formalized as finding a mapping $f : X \to Y$ that minimizes a certain distortion measure, such as the sum of squared distances between items in X and their corresponding images in Y.

There are two main types of machine learning algorithms : supervised algorithms and unsupervised algorithms. Supervised algorithms are used when we have labeled data, which means training data with examples of expected outcomes. The algorithm learns from these examples to make predictions on new data. For example, in the medical field, a supervised algorithm can learn from a dataset of labeled patient data with breast cancer diagnoses to predict whether a new biopsy is suspicious for breast cancer or not.

On the other hand, unsupervised algorithms are used when we have unlabeled data, which means data without examples of expected outcomes. These algorithms search for patterns or structures in the data without using labeled examples. For example, in the medical field, an unsupervised algorithm can analyze health data of patients to group patients with similar symptoms and discover subgroups of patients sharing common characteristics, which can help doctors better understand and treat certain diseases.

In summary, supervised algorithms are used for prediction of labeled values, while unsupervised

algorithms are used for discovery of hidden patterns or structures in unlabeled data.

3.1.1 Supervised Machine Learning

Supervised learning is a type of machine learning where a learner is trained on a set of labeled dataset, denoted by $D = (x_i, y_i)_{i=1}^n$, where $x_i \in \mathbb{X}$ is a feature vector and $y_i \in \mathbb{Y}$ is the corresponding label. The goal of supervised learning is to learn a function $f : \mathbb{X} \to \mathbb{Y}$ that maps input features to output labels. This learned function is then used to make predictions on unseen examples.

Supervised learning is commonly used in classification, regression, and ranking problems. In classification, the learner is trained to predict the correct class label y for each example x. In regression, the learner is trained to predict a continuous value y for each example x. In ranking, the learner is trained to order a set of items according to a criterion.

An example of supervised learning is the spam detection problem, where the learner is trained on a set of emails labeled as either spam or not spam. The learned function is then used to predict whether new emails are spam or not.

In the following, we will provide an overview of the supervised learning algorithms covered in this thesis, namely the k-nearest neighbors algorithm and linear regression.

3.1.1.1 k-nearest neighbors algorithm

The k-nearest neighbors (k-NN) is a simple and widely used supervised learning algorithm in the field of machine learning. It can be used for classification and regression. In this algorithm, the main idea is to find the k closest neighbors to an input data point and to take their average class label (in the case of classification) or their average value (in the case of regression) as the predicted output.

Mathematically, given a training set $T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ where $x_i \in \mathbb{R}^d$ and y_i is the class label or target value associated with x_i, and a test point x_{test}, the k-NN algorithm works as follows :

1. Compute the distance $d(x_i, x_{test})$ between the test point x_{test} and each training point x_i, where $d(\cdot, \cdot)$ is a distance metric such as Euclidean distance, Manhattan distance, or cosine similarity.

$$d(x_i, x_{test}) = \sqrt{\sum_{j=1}^{d}(x_{i,j} - x_{test,j})^2} \qquad \text{(Euclidean distance)}$$

2. Select the k training points with the smallest distances to x_{test}.

$$N_k(x_{test}, T) = \{x_i \in T \mid d(x_i, x_{test}) \leq d(x_j, x_{test}) \text{ for all } x_j \in T \setminus \{x_i\}\}.$$

3. For classification, assign the class label that occurs most frequently among the k nearest neighbors to the test point x_{test} ; for regression, take the average of the target values associated with the k nearest neighbors as the predicted output for x_{test}.

$$\hat{y} = \begin{cases} \arg\max_y \sum_{x_i \in N_k(x_{test}, T)} [y_i = y] & \text{for classification} \\ \frac{1}{k} \sum_{x_i \in N_k(x_{test}, T)} y_i & \text{for regression} \end{cases}$$

Note that the choice of k is a hyper-parameter that needs to be tuned to achieve optimal performance on a given task and dataset. A larger k can lead to a smoother decision boundary or regression function, but can also result in increased computational complexity and overfitting. Conversely, a smaller k can lead to more complex and possibly overfit models, but with increased risk of noise and outliers affecting the predictions.

3.1.1.2 Linear regression

Linear regression is a method for modeling the relationship between a scalar response variable and one or more explanatory variables (also known as dependent and independent variables). When there is only one explanatory variable, it is called simple linear regression. When there are multiple explanatory variables, the method is called multiple linear regression. It is important to note that this term is different from multivariate linear regression, where multiple correlated dependent variables are predicted instead of a single scalar variable.

Given a training set $T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, the goal of linear regression is to find a linear function $f(x) = w^T x + b$ that best approximates the relationship between the input variables x and the output variable y. Specifically, the linear regression algorithm seeks to find the weight vector $w \in \mathbb{R}^d$ and bias term $b \in \mathbb{R}$ that minimize the mean squared error

(MSE) between the predicted values $\hat{y}_i = f(x_i) = w^T x_i + b$ and the actual values y_i in the training set :

$$\underset{w,b}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

This optimization problem can be solved analytically by computing the closed-form solution for the weight vector and bias term :

$$w = (X^T X)^{-1} X^T Y$$

$$b = \bar{y} - w^T \bar{x}$$

where $X \in \mathbb{R}^{n \times d}$ is the design matrix with each row corresponding to an input variable vector x_i, $Y \in \mathbb{R}^n$ is the target vector with each entry corresponding to the output variable y_i, $\bar{x}$ and $\bar{y}$ are the sample means of the input variables and output variable, respectively, and $(\cdot)^{-1}$ denotes the matrix inverse.

Once the weight vector and bias term have been computed, the linear regression model can be used to make predictions on new input variables x_{test} by computing $f(x_{test}) = w^T x_{test} + b$.

Note that linear regression assumes a linear relationship between the input variables and the output variable, and may not be suitable for datasets with nonlinear relationships. Additionally, regularization techniques such as L1 or L2 regularization can be used to prevent overfitting and improve generalization performance.

3.1.2 Unsupervised Machine Learning

3.1.2.1 k-means clustering

Given a dataset $X = \{x_1, x_2, ..., x_n\}$ where each x_i is a d-dimensional vector, the K-means algorithm seeks to partition the dataset into K clusters, where each cluster is represented by its centroid, or mean vector. The algorithm proceeds as follows :

1. Initialize K centroids randomly, e.g., by selecting K data points from the dataset as the initial centroids.

2. Assign each data point to the nearest centroid based on the Euclidean distance metric.

3. Recompute the centroids as the mean vector of the data points assigned to each cluster.

4. Repeat steps 2-3 until convergence, i.e., until the centroids no longer change or a maximum number of iterations is reached.

More formally, let $C = \{c_1, c_2, ..., c_K\}$ be the set of centroids, and let $y_i \in \{1, 2, ..., K\}$ denote the cluster assignment for data point x_i. The goal of K-means is to minimize the within-cluster sum of squares (WCSS), which is defined as :

$$\text{WCSS}(C) = \sum_{k=1}^{K} \sum_{i:y_i=k} \|x_i - c_k\|^2$$

where $\|\cdot\|$ denotes the Euclidean distance. The K-means algorithm seeks to find the centroids that minimize the WCSS, i.e.,

$$\underset{C}{\text{argmin}}\ \text{WCSS}(C)$$

Note that K-means is sensitive to the initial centroids, and different initializations may lead to different results. To mitigate this issue, multiple runs with different initializations are often performed, and the best clustering result is selected based on the WCSS.

K-means can be extended to handle various variations and extensions, such as weighted K-means, K-medoids, and kernel K-means, among others.

3.2 Introduction to Homomorphic Encryption

Homomorphic Encryption allows any entity (for example, the cloud provider) to operate on private data in encrypted form without ever decrypting it. The goal of HE is to perform operations on plain text while manipulating only ciphertexts. Usually, we must decrypt them and then apply the desired processing to manipulate encrypted data. For example, one widespread use case is outsourcing healthcare data to cloud computing services for medical studies (Figure 3.1).

For some cryptosystems with algebraic structures, some operations are possible. For example, two RSA ciphertexts can be multiplied to obtain the multiplication of the two corresponding plain texts.

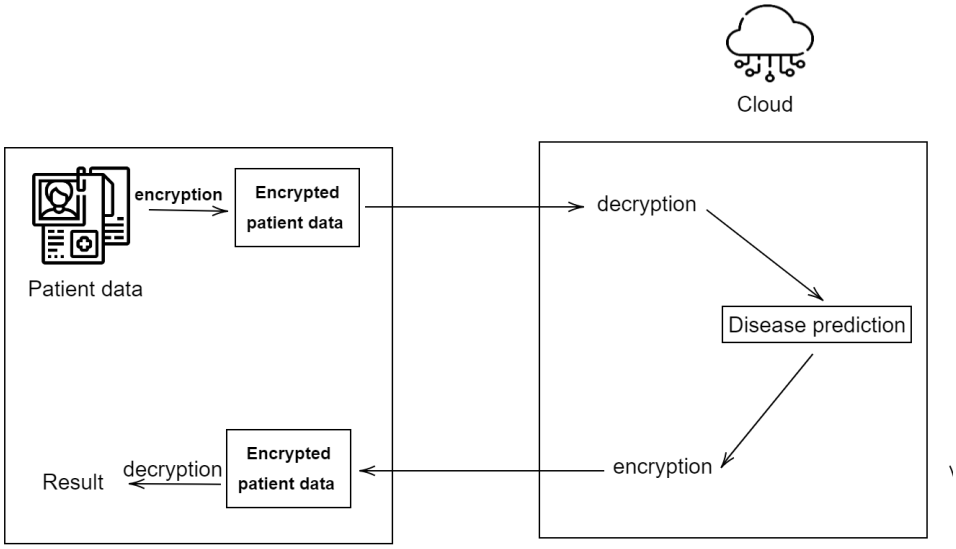


FIGURE 3.1 – "diagram showing how to manipulate encrypted data on a cloud. On the left, the user encrypts the data before sending it to the cloud, on the right, the cloud service has to decrypt the data in order to process it."

We call this property the multiplicative homomorphic property of the "textbook RSA" cryptosystem. Another operation can also be performed on ciphertexts. For example, in the Paillier cryptosystem [9] , we can add two ciphertexts to obtain the addition of the two corresponding plain texts. We call this property the additive property of the "Paillier" cryptosystem. For example, this can be useful when we are interested in e-voting applications to add encrypted votes without knowing the initial vote.

Rivest, Adelman and Dertouzos first introduced the notion of homomorphic encryption in [10]. Building a cryptosystem with both multiplicative and additive properties was a significant problem in cryptography, until the work of Gentry [11]. Gentry proposed a first Fully homomorphic encryption based on ideal lattices. The HE is categorized depending on the number of mathematical operations performed on the encrypted message as following : **Partially Homomorphic Encryption** (PHE), **Somewhat Homomorphic Encryption** (SHE), and **Fully Homomorphic Encryption** (FHE). Over encrypted data, FHE allows arbitrary addition and multiplication. Because any functions may be expressed as Boolean circuits, an encryption scheme capable of performing addition and multiplication can theoretically evaluate any polynomial function[12]. When operations are performed on ciphertexts, the noise increases and too much noise disables accurate decryption. The **bootstrapping** approach is used

by FHE systems to get around this as stated in [11]. Bootstrapping decreases the collected noise, allowing further computation. This procedure can be done as many times as necessary to analyze any particular circuit. However, bootstrapping is computationally costly, so many solutions do not employ it in reality. Therefore, we recommend the reader to refer to [13] for more detailed information on the different homomorphic encryption schemes. We have chosen not to describe the entire functioning of cryptosystems due to the lack of space. Also, selecting secure and efficient instantiations of the underlying cryptographic problem is hard for most of encryption and homomorphic schemes. Therefore, we have chosen to list the most studied schemes by the community of researchers and developers interested in advancing homomorphic encryption.

As usual, new cryptographic proposals need a few years before widespread adoption in the industry, as was the case of Elliptic Curve Cryptography, post-quantum encryption and many other standardization projects.

We are waiting for the standardization results and recommendations of the workshops, which include representatives from industry, government organizations and academia.

This brief review aims to guide readers fast enough, even if they are not cryptography specialists, to the appropriate HE scheme by directing them to the library(ies) where HE is implementable.

3.2.1 HE Schemes

Research in the field of FHE may be classified into four major groups. The first family represents the difficulty based on the lattice reduction problem, which mainly comes from Gentry's seminal work[11]. The second category consists of integer-based methods [14], the hardness of which is based on the Approximate of Greatest Common Divisor (A-GCD) problem [15]. Schemes based on Learning with Error (LWE) [16] and Ring Learning with Error (RLWE) [17], both reducible to lattice problems, constitute the third family. Finally, the Nth-Degree Truncated Polynomial Ring Unit (NTRU) family [18].

All HE schemes have common steps : key generation, encryption, decryption and homomorphic operations on the ciphertexts.

Table 1 summarizes the most implemented and studied schemes by the cryptographic community, and the figure 3.2 give an overvieww of the Homomorphic Encryption timeline.

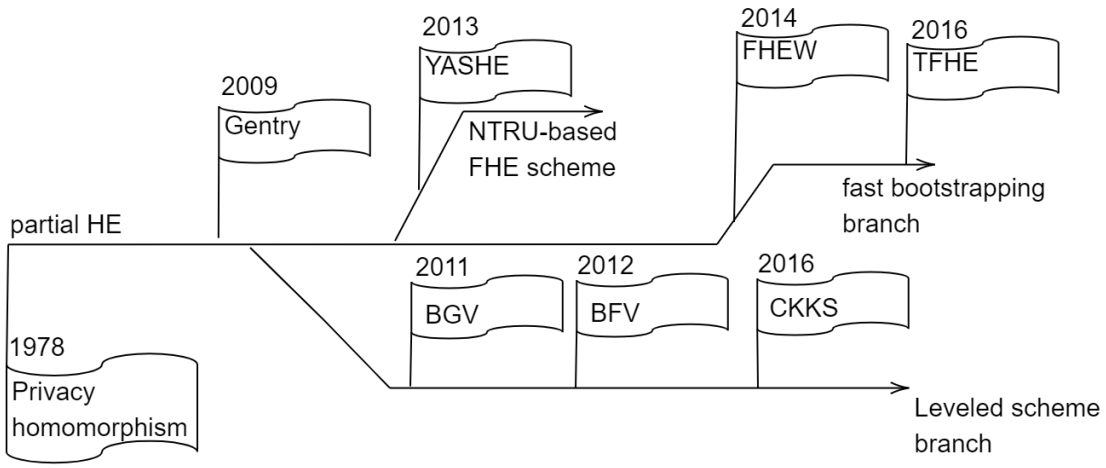


FIGURE 3.2 – Homomorphic Encryption timeline

TABLE 3.1 – Comparison of HE schemes

Operation	**SCHEMES**				
	BFV	**BGV**	**CKKS**	**FHEW**	**TFHE**
Native Add/Sub	✓	✓	✓	✗	✗
Native Mult	✓	✓	✓	✗	✗
Boolean Logic	✗	✗	✗	✓	✓
SIMD	✓	✓	✓	✓	✓

The choice of encryption scheme has a multitude of implications :

— It specifies which operations are possible and, as a result, which types of activation and architectures may be employed.

— It can determine the plaintext space. Messages should be encoded before they may be sent in plaintext. The majority of schemes, including BGV, BFV, only support integers. CKKS can handle real numbers, but TFHE can only handle individual bits.

In the following, we will describe the two schemes used in this thesis, namely TFHE[19] and Paillier's homomorphic encryption[9].

3.2.1.1 Fully homomorphic encryption over the torus : TFHE scheme

TFHE is a fully homomorphic cryptosystem with security based on the (Ring) Learning With Errors problem [20]. It is based on the FHEW cryptosystem [21], but it features much faster bootstraps

thanks to the use of binary secrets [22]. In this section, we review the concepts of TFHE necessary for the understanding of this paper.

Let A be a set, we denote by $A_{n,q}$ the set of vectors with n elements in A modulo q and by $A_{N[X],n}$ the set of vectors of n polynomials modulo $(X^N + 1)$. If omitted, $n = 1$ and $q = \infty$. The Real Torus $T = \mathbb{R}/\mathbb{Z}$ is the set of real numbers modulo 1 and $B = \mathbb{Z}_2$ is the set of binary numbers $0, 1$. TFHE defines three types of ciphertexts, which we summarize below as samples of zero.

— **TLWE Sample :** A pair $(a, b) \in T^{n+1}$, where $b = \langle a, s\rangle + e$. The vector a is uniformly sampled from T^n, the secret key s is uniformly sampled from B^n, the error $e \in T$ is sampled from a Gaussian distribution with mean 0 and standard deviation σ, and $\langle , \rangle$ denotes the inner product.

— **TRLWE Sample :** A pair $(a, b) \in A_{N[X],k+1}$, where $b = a \cdot S + e$. The vector a is uniformly sampled from $A_{N[X],k}$, the secret key S is uniformly sampled from $B_{N[X],k}$, and the error $e \in A_{N[X]}$ is a polynomial with random coefficients sampled from a Gaussian distribution with mean 0 and standard deviation σ.

— **TRGSW Sample :** A vector of k TRLWE samples.

Encryption : To encrypt a message $m \in T$ (TLWE) or $m \in A_{N[X]}$ (TRLWE), we simply add $(0, m)$ to a fresh sample of zero. We denote by $c \in T(R)LWE_s(m)$ the T(R)LWE sample c that encrypts m with keys s. To ease the notation, we consider each key has its attached set of parameters. A message $m \in A_{N[X]}$ can also be encrypted in TRGSW samples by adding $m \cdot H$ to a TRGSW sample of zero, where H is a gadget decomposition matrix. We do not use TRGSW samples in our algorithms and, therefore,We will give into the details only as and when required.

Decryption : To decrypt a sample, we first calculate its phase (message + error) : $\varphi(c) = b - \langle a, s\rangle$. Considering approximate computing, the phase might be a good enough approximation for the message (depending on the error variance). For exact computing, we need to remove the error, and we do so by rounding the phase to the nearest valid value for messages. This requires us to define a set of valid messages over the Torus. The rounding procedure fails if the error

Arithmetic

We add two ciphertext (TLWE or TRLWE) samples $c_1 = (a_1, b_1)$ and $c_2 = (a_2, b_2)$ by simply adding their terms : $c_1 + c_2 = (a_1 + a_2, b_1 + b_2)$. The multiplication between a ciphertext $c_1 = (a_1, b_1)$ and a scalar cleartext $z \in \mathbb{Z}$ (or $z \in \mathbb{Z}[N[X]]$ for TRLWE) is a direct result from the addition :

$c_1 \cdot z = (a_1 \cdot z, b_1 \cdot z)$. TFHE does not support multiplications between $T(R)$LWE samples. To be fully homomorphic, it relies on external products between TRGSW and TRLWE samples. In this case, we first decompose the TRLWE sample in TRLWE samples using a Gadget decomposition algorithm [GMP19]. Then, we perform an inner product between the decomposed TRLWE and the TRGSW sample (which already is a vector of TRLWE samples).

3.2.1.2 Additive Paillier cryptosystem

This section is inspired from [23]. Proposed in 1999 by Paillier [9], the Paillier cryptosystem is based on the problem that computing nth residue classes is computationally intensive. The nature of the algorithm allows for homomorphic addition operations to produce the current answer once decrypted. The key generation for Paillier Cryptosystem is given in **Algorithm 1**.

Algorithm 1 Paillier cryptosystem key generation algorithm

1: Select two large prime numbers p and q where $\gcd(pq, (p-1)(q-1)) = 1$
2: Calculate $n = pq$
3: Calculate $\lambda = \text{lcm}(p-1, q-1)$
4: Select g as a random integer where $g \in \mathbb{Z}^*_{n^2}$
5: Define $L(x) = \frac{x-1}{n}$
6: Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse :
7: $u = (L(g^\lambda \bmod n^2))^{-1} \bmod n$
8: Public Key $= (n, g)$
9: Private Key $= (\lambda, u)$

To encrypt a message, the message is used as the exponent for g, then a random value is raised to the other public key value n, as shown in **Algorithm 2**. This produces a cipher value in modulo n^2.

Algorithm 2 Paillier cryptosystem encryption algorithm

1: Encrypt a message M where $M \in \mathbb{Z}n$
2: Select r as a random integer where $r \in \mathbb{Z}n^{2*}$
3: Calculate $c = g^M \cdot r^n \bmod n^2$

Decryption is again a simple equation, and is given in **Algorithm 3**. Note that the definition for $L(x)$ was given with key generation.

The proof of the encryption and decryption will be now given to show how the public and private key values cancel each other out. This is important because it will help to show why the Paillier

Algorithm 3 Paillier cryptosystem decryption algorithm

1: Decrypt a message c where $c \in \mathbb{Z}_{n^2}^*$
2: Calculate $m = L(c^\lambda \bmod n^2) \cdot u \bmod n$

Cryptosystem can support homomorphic addition operations.

1. $m = L(c\lambda \bmod n^2) \times u \bmod n$
2. $m = L(c\lambda \bmod n^2) \times (L(g\lambda \bmod n^2))^{-1} \bmod n$
3. $m = \prod_{i=1}^{k} L((c\lambda \bmod n^2)^{a_i} \bmod 2) \bmod n$, where k is the number of bits in n
4. $m = (1+n)^{L(c\lambda \bmod n^2)} \cdot (1+n)^{-L(g\lambda \bmod n^2)} \bmod n$ (By Lemma 10 in Paillier (1999))
5. $m = (1+n)^{L(c\lambda \bmod n^2)} \cdot (1+n)^{-L(g\lambda \bmod n^2)} \bmod n$
6. $m = [c]g \bmod n$
7. $m = m \bmod n$ (Because $c = \frac{(g^m \cdot r^n) \bmod n^2}{(g^n \bmod n^2)^r \bmod n^2}$ (Paillier, 1999))

If an addition operation is desired to be computed on the encrypted data, it is actually a multiplication operation that needs to be used. This is because the message is encrypted as an exponent. Therefore to add exponents, a multiplication operation needs to be computed on two values of the same base, which in this case is g. Note that because the r_0 and r_1 values are random, they can be combined to form another random value r.

3.2.2 HE Librairies

There exist several open-source libraries for the implementation of the HE scheme. They provide key generation, encryption, decryption, and homomorphic operations for each scheme ; library APIs frequently include additional features for maintaining and manipulating ciphertexts. Even though there is a lack of technical interoperability, but also a lack of conceptual interoperability ; for example, even libraries that use the same scheme can provide surprisingly different results. The ongoing standardization efforts attempt to develop a unified view of the most popular schemes.

3.2.3 FHE Restrictions

Current HE methods have a significant restriction. They cannot support division operations and comparisons easily, such as the equality/inequality test. Number comparison and sign determination are critical processes for MLaaS.

TABLE 3.2 – Overview of existing FHE librairies : CPU-targeting (top) and GPU-targeting (bottom)

Name	**Input language**	**Supported schemes**				
		BFV	**BGV**	**CKKS**	**FHEW**	**TFHE**
HE-CPU-TARGETING						
Concrete	Rust	✗	✗	✗	✗	✓
FHEW	C++	✗	✗	✗	✓	✗
FV-NFlib	C++	✓	✗	✗	✗	✗
HEAAN	C++	✗	✗	✓	✗	✗
HElib	C++	✓	✓	✓	✗	✗
lattigo	Go	✓	✗	✓	✗	✗
PALISADE	C++	✓	✓	✓	✓	✓
SEAL	C++, .NET	✓	✓	✓	✗	✗
TFHE	C++	✗	✗	✗	✗	✓
HE-GPU-TARGETING						
cuFHE	C++, Python	✗	✗	✗	✗	✓
nuFHE	C++, Python	✗	✗	✗	✗	✓

3.3 State of the Art of Privacy-Preserving in Machine Learning (PPML) : HE-based solutions

This section presents an in-depth exploration of the current state of research surrounding the contributions of this thesis. Drawing from a previously published book chapter as a reference [24].

In recent years, the use of third-party infrastructures has gained popularity in the field of machine learning, as it offers advantages such as reduced resource constraints and simplified complexity. However, this approach also brings along concerns related to the privacy of sensitive information. In the context of my thesis, which focuses on building a privacy-preserving framework for machine learning techniques, it is crucial to identify the key privacy requirements. These requirements may include but are not limited to, ensuring data confidentiality, protecting against unauthorized access, minimizing data exposure, and complying with relevant data protection regulations. Addressing these privacy concerns is essential to develop a robust framework that maintains the privacy of sensitive information while utilizing third-party infrastructures to overcome resource and complexity challenges.

— Input privacy : Only the real data owner should have access to the input.

— Output privacy : The output/result of the ML methods' assessment is not permitted to be known by the cloud server.

— Model privacy : A private machine learning model that is also an asset should not be shared with anyone except its owner.

Another approach is to look if the privacy-preserving framework targets the learning phase or the inference phase of the machine learning algorithm.

Depending on the framework we want to design, we have to use privacy-preserving technologies. The leading privacy Preserving Machine Learning techniques are the following :

— **Multi-party Computation** : These methods involve one or more trusted parties that can be used to outsource specific computations by the algorithm owner.

— **Differential Privacy** : These methods rely on data randomization and perturbation. Because it affects the information, this method has the drawback of negatively influencing the model's performance.

— **Federated Learning** : Federated learning is a machine learning setting where many clients collaboratively train a model under the administration of a central server while keeping the

training data local.

— **Garbled Circuit (GC)** : Garbled Circuit, also known as Yao's garbled circuit, is an underlying technology of secure two-party computation initially proposed by Andrew Yao [? ? reference]. GC provides an interactive protocol for two parties (a garbler and an evaluator) obliviously evaluate an arbitrary function represented as a Boolean circuit.

— **Homomorphic Encryption** : An encryption that allows performing operations over encrypted data. See section 3.2 for more details.

— **Hybrid PPML Techniques** : In addition to the above-mentioned single-protocol PPML, some commonly used frameworks typically use hybrid protocols, which combine two or more protocols by making use of the advantages and avoiding the problems of each. For example, the basic idea behind the mixed protocol that combines HE and GC is to compute operations that have an efficient representation as Arithmetic circuits (e.g., additions and multiplications) using HE and operations that have an efficient representation as Boolean circuits (e.g., comparisons) using GC. However, converting between share systems is not simple, and the charges are very high. Furthermore, various frameworks integrate MPC with Differential Privacy.

We are interested here in Machine Learning Research Using Homomorphic Encryption "HEML". According to this Bibliometrics [25], the number of papers on HEML has constantly been rising since 2009. Each year from 2005 to 2015, fewer than 100 HEML papers were published. However, the number of publications per year increased significantly after 2015, reaching between 200 and 500 in recent years. This section summarizes recent works and gives many practical applications of homomorphic encryption for privacy-preserving purposes for each machine learning algorithm. We end the section with a summary of the work to ease the reading of this synthesis.

3.3.1 Logistic Regression

Logistic regression is a powerful machine-learning approach that uses a logistic function to model two or more variables. Logistic models are commonly used in the medical community to predict binary outcomes, such as whether a patient requires treatment or whether a disease appears [26]. It has been utilized in applications such as evaluating diabetes patients' medications [27], and social sciences[28].

iDASH is an annual competition that attempts to deploy novel cryptographic methods in a biological environment. Since 2014, genomics and biomedical privacy have been incorporated into iDASH.

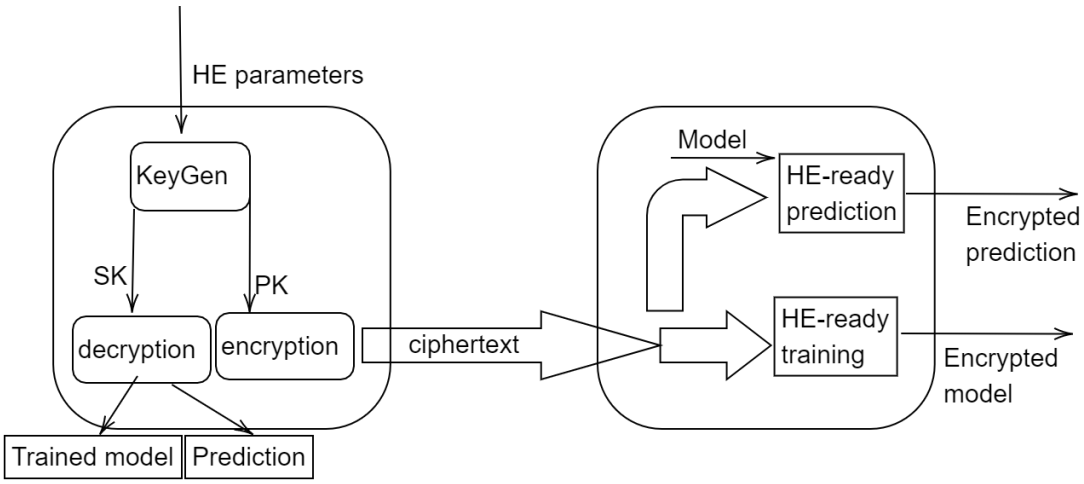


FIGURE 3.3 – HEML scenario

Both the third track of the 2017 iDASH competition [29] and the second track of the 2018 iDASH competition driven the development of homomorphic encryption-based solutions for building a Logistic regression model over encrypted data. The performance of LR training based on homomorphic encryption (HE) has improved significantly as a result of these two competitions.

Homomorphic encryption has been used in much research on training logistic regression models.

Authors, in [30], trained a privacy-preserving logistic regression model using HE ; however, the time complexity of linear HE increases exponentially with the number of parameters.

In [31], the authors used an additive HE scheme and delegated particular challenging HE computations to a trusted client, the authors in this work introduced an approximation to convert the likelihood function into a low-degree polynomial.

The issue of doing LR training in an encrypted environment was discussed by [32]. They used complete batch gradient descent in the training phase, using the least-squares approach to approximate the logistic regression. They also employed the CKKS scheme, which allows for a homomorphic approximation of the sigmoid function.

There is no closed-form solution to logistic regressions, so we must use non-linear optimization methods to find the maximum likelihood estimators of the regression parameters. During training, gradient descent and Newton-Raphson are the most commonly used methods. The Newton-Raphson method requires matrix inversion, and most HE schemes do not natively support division and matrix

inversion. On the other hand, Gradient descent does not require the division operation and so is a better candidate for homomorphic logistic regression.

Although the gradient descent approach appears to be better suited for homomorphic evaluation than other training methods, some technical issues remain for implementation. The sigmoid function is the most challenging to evaluate since existing homomorphic encryption techniques only allow the evaluation of polynomial functions, so Taylor polynomials have been widely employed for sigmoid function approximation [33, 34].

For implementation and performance of Private Logistic Regression (He-based solutions), see Table 3.3.

3.3.2 Naive Bayes and Decision Trees

Naive Bayesian classification is a simple probabilistic Bayesian classification based on Bayes' theorem. It uses a naive Bayesian classifier, or naive Bayes classifier, belonging to the family of linear classifiers. In [35], the authors propose a privacy-preserving Naive Bayes classification algorithm. A client learns the classification of her data point X in their model without knowing the classification model or disclosing any information about her input. The model's estimated parameters are encrypted and transferred to a cloud server. The authors use two partially homomorphic encryption schemes, Quadratic Reciprocity [36] and Paillier [9], in the same work [35] implements a privacy-preserving strategy for three algorithms, and one of those is decision trees. This work has shown that polynomials may be utilized to express decision trees. The decision tree node values must be compared to the evaluation data, and the outputs must be used to construct the polynomial, yielding the evaluation results.

For the implementation and performance of Private Naive Bayes and Decision tree (He-based solutions), see Table 3.4.

3.3.3 K-nearest neighbors

The k-Nearest Neighbors (k-NN) is a simple method that can handle continuous, categorical, and mixed data. Furthermore, as a non-parametric method, k-NN can be relevant for many data structures as long as the number of observations is sufficiently large. In addition, for a predefined number of neighbors k, the model does not require any training step since the prediction for a new

TABLE 3.3 – Summary of main works on Private Prediction for Logistic Regression

REF	**HE scheme / Type**	**Plateform**	**Evaluation Time**	**Accuracy**	**Datasets**
Logistic Regression					
[16]	[1] /LHE	-	-	82.89%	Dataset SPECT -267 instances -23 features
[17]	[1] /LHE	2.60 GHz x 2 CPU, 128 GB RAM.	-	73.7%	SPECTF heart dataset -267 instances -44 features)
[17]	[1] /LHE	2.60 GHz x 2 CPU, 128 GB RAM.	-	80.7%	Pima diabetes dataset -768 instances -8 features
[18]	CKKS / LHE	intel Xeon 2.3 GHz processor with 16 cores and 64GB of RAM,	131min	86.03%	Edinburgh -1253 instances -10 features
[18]	CKKS / LHE	intel Xeon 2.3 GHz processor with 16 cores and 64GB of RAM,	101min	69.30%	Lbw -189 instances -10 features
[18]	CKKS / LHE	intel Xeon 2.3 GHz processor with 16 cores and 64GB of RAM,	265min	79.23%	Nhanes3 -15649 instances -16 features
[18]	CKKS / LHE	intel Xeon 2.3 GHz processor with 16 cores and 64GB of RAM,	119min	68.85%	Pcs - 379 instances - 10 features
[18]	CKKS / LHE	intel Xeon 2.3 GHz processor with 16 cores and 64GB of RAM,	109min	74.43%	Uis -575 instances, -9 features
[19]	YASHE / LHE	Intel Core i7-3520M at 2893.484 MHz	-	-	Heart Disease Framingham -4,000 instances - 15 features
[20]	linearly homomorphic encryption	Amazon EC2 c4.8xlarge machines running Linux, with 60GB of RAM each.	149.7sec	98.62%	MNIST dataset -60 000 instances -784 features

TABLE 3.4 – Summary of main works on Private Prediction for Naive Bayes and Decision Tree

REF	**HE scheme / Type**	**Plateform**	**Evaluation Time**	**Accuracy**	**Datasets**
Naive Bayes					
[21]	[1] + [22] /LHE	two Intel Core i7 (64 bit) processors for a total 4 cores running at 2.66 GHz and 8 GB RAM.	479 ms	-	Breast Cancer -2 classes -9 features
[21]	[1] + [22] /LHE	two Intel Core i7 (64 bit) processors for a total 4 cores running at 2.66 GHz and 8 GB RAM.	1415 ms	-	Nursery -9 classes -5 features
[21]	[1] + [22] /LHE	two Intel Core i7 (64 bit) processors for a total 4 cores running at 2.66 GHz and 8 GB RAM.	3810 ms	-	Audiology -14 classes -70 features
Decision tree					
[21]	[1] + [22] /LHE	two Intel Core i7 (64 bit) processors for a total 4 cores running at 2.66 GHz and 8 GB RAM.	239 ms	-	Nursery
[21]	[1] + [22] /LHE	two Intel Core i7 (64 bit) processors for a total 4 cores running at 2.66 GHz and 8 GB RAM.	899 ms	-	ECG

observation is obtained by :

— Identifying the k nearest neighbors (according to a given distance)

— Computing the majority class among them (for a classification problem) or by averaging values (for a regression problem).

Homomorphic encryption has already been investigated by various authors for k-NN [37, 38, 39,

6].

HE has been recently investigated by various authors for k-NN [37, 38, 39, 6].

In [37], the authors suggested a homomorphic additive encryption scheme [40]. They investigated privacy preservation in an outsourced k-NN system with various data owners. The untrusted entity securely computes the computations of distances by using HE. However, the comparison and classification phases require interactions. Given that the computational and communication difficulties scale linearly, they admit that the method may not be practical for massive data volumes. The cost of communications between the entities is also a limitation in the deployment of this work [38].

Other authors, in [39], used an "asymmetric scalar-product-preserving encryption" (ASPE). However, the client has the ciphertext, and the server can decrypt it. The proposed solution is vulnerable to Chosen Plaintext Attacks, as stated by [41].

Recently, authors in [6] proposed a secure k-NN algorithm in quadratic complexity concerning the size of the database completely non-interactively by using a fully homomorphic encryption [19]. However, they assume that the majority vote is done on a clear-text domain, which is a significant security flaw that we will address here. Doing a majority vote on a clear-text domain imposes interaction between entities, which causes information leakage.

The authors of [37] suggested a Homomorphic additive encryption scheme [9]. They investigated the privacy preservation in an outsourced k-NN system with various data owners. The untrusted entity securely computes the distances by using HE. However, the comparison and classification phases require interactions. Given that the computational and communication difficulties scale linearly, they admit that the method may not be practical for massive data volumes. The cost of communications between the entities is also a limitation in the deployment of this work [38].

Recently, [6] proposed a secure k-NN algorithm in quadratic complexity concerning the size of the database completely non-interactively by using a fully homomorphic encryption [22]. However, they assume that the majority vote is done on a clear-text domain, which is a significant security flaw that we will address here. Doing a majority vote on a clear-text domain imposes interaction between entities, which causes information leakage. For implementation and performance of Private k nearest neighbors (HE-based solutions), see the table 3.5

TABLE 3.5 – Summary of main works on Private K-nearest neighbors

REF	**HE scheme / Type**	**Plateform**	**Evaluation Time**	**Accuracy**	**Datasets**
K-nearest neighbors					
[23]	[1] /LHE	-	-	97.85%	Cancer 1 (9 features)
[23]	[1] /LHE	-	-	96.49%	Cancer 2 -569 instances, -30 features)
[23]	[1] /LHE	-	-	81.82%	Diabetes
[23]	[1] /LHE	-	-	97%	MNIST dataset -60 000 instances -784 features
[26]	[27] /FHE	Intel Core i7-6600U CPU	11.6 min	94.8	MNIST dataset -60 000 instances -784 features

3.3.4 Neural Networks and Deep Learning

Deep learning is one of the most sophisticated techniques in machine learning, and it has received a lot of attention in recent years. It is presently employed in a variety of areas and applications, including pattern recognition, medical prediction, and speech recognition. Deep learning experiences are enhanced significantly by utilizing strong infrastructures such as cloud data centers and implementing collaborative learning for model training. However, this compromises privacy, particularly when sensitive data is processed during the training and prediction stages, as well as when the training model is disseminated. In this section, we discuss known privacy-preserving deep learning algorithms based on homomorphic encryption, we present recent challenges concerning the intersection of HE cryptosystems and Neural Networks models, as well as methods to overcome limitations.

HE cannot be used naively in neural networks algorithms. There are a lot of challenges and restrictions that must be overcome. The constraints differ according to the scheme, however, several common issues emerge in most systems. The learning and inference phases of the deep learning algorithm can be distinguished.

3.3.4.1 Privacy preserving deep learning : Private training

Several techniques have been proposed ; they consider **collaborative training**, in which the training is performed collaboratively between different participants, or **individual training**, in which the training is performed by a single participant, such as a client who wants to use a cloud to train its model.

Authors of [42] proposed a solution in **collaborative learning** mode, where participants send the calculated encrypted local gradients to the cloud after each iteration of local training, starting with the initial weights obtained from the cloud. To ensure homomorphic ciphertext integrity, each participant uses a unique TLS/SSL secure channel. The cloud then updates the encrypted global weights vector, which the participants download. The approach theoretically achieves the same accuracy as standard asynchronous SGD, whereas evaluations show that MLP and CNN reach 97 percent and 99 percent accuracy, respectively. In terms of efficiency, an overhead in communication and calculation was seen ; however, the authors considered it negligible. In addition, the accuracy/privacy trade-off might be adjusted to efficiency/Privacy, allowing the precision to be preserved while maintaining Privacy.

The privacy-preserving back-propagation technique described in [43] is based on BGV fully homomorphic encryption and Maclaulin polynomial approximation of the sigmoid activation function. The client encrypts input data and configured parameters before uploading them to the cloud, which executes one loop. The client downloads and decrypts the findings before updating its local model. It then encrypts and sends the updated parameters back to the cloud, which repeats the process. This method is repeated until the maximum error threshold or the number of iterations is achieved. Although BGV encryption provides for the protection of private data throughout the learning process, it does need the approximation of the activation function. This might lead to a drop in accuracy. In terms of efficiency, while the solution achieved a two times greater efficiency, i.e., 45 percent of the training time of the standard model, it experienced compute and communication costs due to the encryption-related overhead.

[44] employs the Taylor theorem to estimate the sigmoid activation function polynomially. The evaluation findings revealed a reduction in accuracy for both classification and prediction tasks. However, the authors proposed adding additional Taylor series terms to decrease classification loss, raising the BGV encryption level, resulting in poor performance. The method could achieve 2.5 times greater classification efficiency and two times higher overall efficiency in learning time.

[45] described a more recent solution based on encryption. A client who wants to participate to the model's training transmits its data encrypted using the Paillier scheme [9] to the server, which performs all possible neural network calculations except non-linear activation functions. To continue execution, the encrypted weighted sums before each activation function are provided to the client, who will be in charge of performing the calculation. The result is then encrypted again and sent back to the server.

3.3.4.2 privacy preserving deep learning : Private inference

Fully homomorphic encryption is deployed in a line of research that performs private classification of encrypted data using a neural network that has been trained using plain data.

[46] is the first solution for Privacy-preserving deep learning for inference, developed by Microsoft Research. The approach is based on the YASHE a (Leveled Homomorphic Encryption) LHE scheme. The user encrypts their data and sends it to the cloud, which runs the model and returns an encrypted prediction. It has been demonstrated that the YASHE scheme is vulnerable to subfield lattice attack [47]. To make the network compatible with homomorphic encryption, max-pooling is replaced by a scaled-mean pool function, and activation functions are approximated by the square function, which is the lowest-degree non-linear polynomial function. According to the authors, these adjustments should preferably be considered during training on unencrypted data. For example, the solution could achieve 99% accuracy and 59000 predictions per hour on a single PC for the MNIST dataset.

To overcome the heavy computation cost of HE, a dual cloud model was proposed [48], in which two clouds, A and B, collaborate to generate classification results in a secure environment. Cloud A operates the neural network on private data encrypted by the client with Paillier cryptosystem [9], but delegates activation function computations to the cloud B since they share a key. The technique is repeated until the final layer is reached. Client A then protects the final output with a random salt from cloud B, which uses the softmax function and sends the final encrypted result to the client. A theoretical scenarios-based security and accuracy study demonstrated how the approach successfully defends against potential threats.

[49] suggested a method for classification problems over the CNN model based on BGV an FHE scheme. The combination of polynomial approximation and batch normalization is the major technological breakthrough. During the training phase, a batch normalization layer is introduced before each

ReLU layer to avoid excessive accuracy deterioration, and max-pooling is replaced by average-pooling, which is more FHE-friendly and has a small overhead.

Prior to each ReLU, a batch normalization layer is introduced with a low-degree (2) polynomial approximation. When the model is complete, the user encrypts its private data and sends it to the model, carrying out the specified analysis. The evaluation findings revealed that the solution has a short running time, with comparable performance, as if there was no privacy.

[50] attempt to use HE for deep learning problems. They provide methods by using low degree polynomials to approximate the most generally used neural network activation functions (ReLU, Sigmoid, and Tanh). This is a critical step in the development of effective homomorphic encryption methods. They then train convolutional neural networks using those approximation polynomial functions before implementing the models over encrypted data and evaluating its performance.

[51] suggest a recent homomorphic encryption-based approach. The user encrypts their personal information and transmits it to the server for prediction. The Paillier scheme accelerates linear, convolutional, and pooling transformations. The authors chose ReLU as the activation function and suggested, rather than utilizing polynomial approximation, an interactive protocol between the client and the server for its computation. The user gets the ReLU input data, decrypts it, and communicates the positivity or negativity of this input to the server, enabling the server to calculate the output. The evaluation findings revealed that the solution could reach near model accuracy in plain text and was similar to Cryptonet [46]. In terms of efficiency, the approach saves a significant amount of time.

Recently, authors in [52] have resulted in considerable improvements by using the scheme TFHE [22]. FHE methods permit unrestricted encrypted operations and give accurate polynomial approximations to non-polynomial activation functions using a programmable bootstrapping technique, an extension of the bootstrapping technique that allows resetting the noise in ciphertext to a fixed level while—at the same time—evaluating a function for free.

3.3.5 Clustering

Clustering is an unsupervised machine learning problem that automatically identifies natural grouping (clusters) in data.

A clustering algorithm can be collaborative or individual. In both cases, a model can be based on

a server, and the calculations are exclusively performed on the server or assisted by a server. In this case, some calculations are delegated to the server.

Three models can be found in the literature :

1. data comes from several parties, and these parties collaborate to train a clustering model.
2. single party that holds the data but not the computational resources needed to perform the calculations. The data is outsourced to perform clustering.
3. data comes from multiple parties and is paired to build a shared database. Then the data is outsourced to perform clustering.

Cases 2 and 3 are similar ; this case is called "outsourced clustering". The first case is called "distributed clustering". Plenty of clustering algorithms have been seen in the privacy-preserving framework : k-means, k-medoids, GMM, Meanshift, DBSCAN, baseline agglomerative HC BIRCH and Affinity Propagation. Among them, k-means has been intensively studied. In what follows, we focus only on the works that use homomorphic encryption.

3.3.6 Collaborative clustering

In the case of collaborative clustering, several parties own data and want to collaborate to get good quality clustering without disclosing the information contained in the data. This case has been extensively studied in two parts. [53] considered the case where two parties with limited computational resources would like to run k-means by outsourcing the computations to the cloud. Both parties will have a result based on both datasets. In this case, one party's data should be kept confidential from the cloud and the other party. The authors used two schemes to propose a solution : the Liu encryption and the Pallier encryption. Each party encrypts the data and sends it to the cloud. The cloud performs calculations and comparisons based on additional information about both parties. To recalculate the centers, the cloud sends the sum of all the vectors to both parties, and the parties use a protocol to compute the new centers. The authors [54] propose a protocol to perform secure k-means in the semi-honest model. In this work, the Pallier scheme has been used. The computation of the Euclidean distance requires interaction with the data owner to perform the multiplications. The comparison is performed using bit-by-bit encryption.

The authors [55] studied clustering using the k-medoids algorithm applied to intrusion detection. Multiple organizations collaborate to perform clustering and have better results without sharing the

content of this information in the clear. In addition, the system relies on a semi-honest party to perform clustering using Pallier encryption. The k-medoid algorithm requires more complex operations than addition. This requires interactions between collaborators to decrypt this data at runtime and thus perform the operations.

3.3.7 Individual clustering

A clustering is individual if only one person has data and she wants to have the results of the clustering of this data. Most of the works interested in this kind of clustering require an intermediate decryption step.

The authors [56] demonstrate a solution to perform k-means using a collaboration between the client and a server. They used the BV scheme [57]. In this work, they proposed three variant solutions. Each solution takes as input a dataset of dimension nxd, an integer k which denotes the number of clusters and a threshold of iterations. The algorithm returns a matrix of dimension kxd that indicates the cluster centers. In the first variant, the computation of the centers and the assignment are done at the client level, which implies that the client performs a lot of computations (only the distances are computed at the server level). In the second variant, the client performs the comparisons and the division. At the same time, the server calculates the distances and the assignment of the points, then the sum to calculate the new centers. This variant induces an information leakage on how the points are distributed on the clusters. Finally, a third variant tries to solve the information leakage problem by returning an encrypted assignment vector of a point instead of the clear assignment. The authors [58] propose a method for k-means that limits interaction with the data owner using the concept of "Updatable Distance Matrix (UDM)". The latter is a 3D matrix whose first two dimensions equal the number of data in the dataset, and the third dimension equals the number of attributes. Each cell in the matrix is initialized to the difference between the attributes of the data vectors. The idea is to save the encrypted data and the UDM matrix to a third party. This matrix is updated at each iteration of k-means using an offset matrix obtained by calculating the difference between the new and current centers. This method is expensive in terms of time and memory to store the UDM matrix.

The authors [59] have tried an exact implementation of k-means that requires no intermediate decryption. Instead, the method relies on building a logic circuit to perform k-means using TFHE. From a theoretical point of view, the method gives equivalent results to the plaintext version. However,

this method is not feasible ; with two dimensions and 400 points, the execution time has been estimated at 25 days.

The authors [60] also propose a solution that focuses on k-means. In this solution, the BGV scheme [61] is used. The authors notice that deciphering the intermediate steps at the client level is a costly operation. The proposed solution relies on using a third party as a trusted entity to decrypt the intermediate results. A private key equivalent (but different) to the owner's and a switch matrix are generated to be used by the trusted server. The proposed solution is considered secure in a semi-honest model but not in the malicious case.

3.4 Conclusion

In concluding this chapter, we undertook an extensive exploration of the critical topic of Privacy-Preserving Machine Learning (PPML). Our discussion began with a detailed overview of homomorphic encryption and an introduction to the fundamental principles of machine learning, setting a solid foundation for the in-depth study of PPML. This was followed by a focused examination of the current best practices and models most commonly used in PPML. This analysis not only highlighted the latest advancements in this rapidly evolving field but also emphasized the ongoing challenges in optimizing privacy within machine learning models, offering a clear and comprehensive understanding of the state of PPML today.

In Chapter 3, we introduce a multi-cloud platform that integrates public, private, and managed clouds into a single user interface, with the goal of enhancing data privacy and availability.

Chapter 4 presents a comprehensive exploration of the application of homomorphic encryption to the k-nearest neighbors (k-NN) algorithm. This includes a practical implementation of the algorithm using homomorphic encryption and a meticulous evaluation of its feasibility on diverse datasets.

Chapter 5 focuses on our original contribution to the field, which is the application of homomorphic encryption to the k-means clustering algorithm. The performance of this approach is rigorously evaluated on various datasets to assess its effectiveness.

In Chapter 6, we pursue a novel research direction by investigating the combination of homomorphic encryption with differential privacy (DP) techniques to further enhance the privacy guarantees of machine learning models.

Finally, Chapter 7 provides a conclusive summary of the research findings and proposes potential avenues for future research in this field.

Troisième partie

Contributions

Chapitre 4

Handling security issues by using homomorphic encryption in multi-cloud environment

4.1 Introduction

Taking advantage of the high performance and powerful data processing capabilities of cloud computing technology, externalizing data to the cloud platform is considered an inevitable trend in the digital field today. However, ensuring the security and privacy of data remains a major challenge. To overcome this drawback, a multi-cloud platform is proposed to improve privacy and high availability of data. A multi-cloud platform that integrates public, private, and managed clouds with a single user interface. Cloud-hosted data is distributed among different data centers in a multi-cloud environment based on cloud reliability and data sensitivity. In terms of security, current encryption algorithms are considered to be very efficient, but it requires a lot of resources to handle this, which is expensive and time consuming. In addition, they also make the data impossible to process without first decoding. To be specific, traditional public key encryption requires data to be decrypted before it can be analyzed or manipulated. In contrast, homomorphic encryption is an encryption method that allows data to be encrypted while it is being processed and manipulated. It allows user or a third party, which can be cloud provider, to apply functions on encrypted data without revealing the data's values. In this introductory example, we explore existing multi-cloud-based security solutions using homomorphic encryption to identify open issues and opportunities for further enhancement.

Cloud computing and related technologies are currently attracting a lot of attention from either

research or industry. Michael Armbrust et al [62] define the cloud as "the long-held dream of computing as a utility". The National Institute of Standards and Technology (NIST) gives another definition [63] "cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." The cloud computing paradigm can be described in simple terms as "everything as a service", where data is accessible over the Internet. The most popular ones are Platform as a Service (PaaS), Infrastructure as a Service (IaaS) and Software as a Service (SaaS). The advantages of cloud computing are undeniable, such as lower computing costs, instant software updates, reduced software costs, and unlimited storage. The multi-cloud service is the next generation in cloud computing's development. The necessity to integrate clouds for enhanced processing and storage capability has become an increasingly relevant subject for IT experts as resource requirements seem to be increasing inexorably. Concerns regarding vendor dependency and cloud failure have also been highlighted, both of which might be mitigated by switching to a multi-cloud environment [64]. A multi-cloud approach is a cloud storage architecture that creates a virtual cloud storage system by combining several cloud storage providers. The data to be saved is divided into separate blocks and redundantly distributed to numerous cloud storage providers. However, collaboration with multiple clouds raises security concerns such as increased attacks, loss of control over data and data privacy issues. To avoid this situation and protect data privacy in multi-cloud environments, one possible solution is to use fully homomorphic encryption (FHE) to ensure privacy. The goal behind FHE is to allow anyone to use encrypted data to perform useful operations without accessing the encryption key. In particular, this concept has applications to improve cloud computing security. In case a user wants to store sensitive, encrypted data in the cloud, but does not trust her cloud provider or is at risk of an Intruder breaking into her cloud account or application, FHE provides a way to pull, search, and manipulate data without having to allow the cloud service provider access to the data.

Homomorphic encryption is an encryption scheme that allows computations on encrypted data, yielding an encrypted result. When decrypted, this result produces the same outcome as computations performed on the original, unencrypted data. The aim of this work is to offer an architectural framework to aid in securing data sharing processes in a multi-cloud context. Specifically, this initial work highlights methods for achieving secure data sharing through the application of homomorphic

encryption.

In the next section, we will survey the state of the art in homomorphic encryption and explore the main challenges in applying it to real-world multi-cloud architectures. Section 3 provides a detailed examination of the privacy challenges in multi-cloud computing and how homomorphic encryption can address them. Our main contribution, which introduces a new architecture designed for multi-cloud electronic medical records, is presented in Section 4. Finally, the conclusions are summarized in Section 5.

4.1.1 Related works

To enhance security and privacy in cloud environments, numerous studies have explored various encryption methods to develop optimal solutions. Cloud computing security encompasses two domains : security issues encountered by cloud providers (organizations that offer software, platforms, or infrastructure services) and those faced by their customers. Innovations and new technologies are extensively applied in both domains. Therefore, in this section, we conduct a review of the state of the art in different forms of cloud security, aiming to identify the advantages and deficiencies of current approaches.

In [65], the authors bring a solution for the K-nearest neighbors (k-NN) algorithm with a homomorphic encryption scheme (called TFHE). The proposed solution addresses all stages of k-NN algorithm with fully encrypted data, including the majority vote for the class-label assignment. Unlike existing techniques, the solution does not require intermediate interactions between the server and the client.

In the context of Multi-Cloud Computing (MCC), the authors in [66] discuss the security implications for mobile users in a multi-cloud computing environment and the advantages it offers them. They highlight the security issues inherent in Mobile Cloud Computing and present the main reasons for transitioning to a multi-cloud approach. Similarly, the concept of a 'light token' was introduced in [67] to enhance security for mobile users within the MCC environment. In this approach, the authors propose a novel attestation schema that builds on existing attestation mechanisms as well as traditional encryption methods. Although the stability of the algorithm has been verified, this approach relies on trusted parties, which poses challenges in controlling data and privacy.

With other approaches, Fabian et al. [68] suggested an architecture for exchanging health care information using Attribute Based Encryption and cryptographic secret sharing. This technique does not ensure the data integrity or efficiency of the full process, which includes uploading, file slicing, and group sharing, among other things. In [69], the authors present a security platform that allows user authentication and data encryption. The platform uses the properties of homomorphic encryption to generate a robust electronic signature. Then to improve the authentication mechanism, the verification tasks are distributed over different virtual machines so that an attacker can never recover or intercept passwords or other personal information of the data subject. In the work of Zibouh et al.[25], a multi-cloud architecture has been proposed with fully homomorphic encryption by using the gentry scheme [70] to enhance the performance and the time of data processing. However if the size of the file increases, computation overhead arises.

4.1.2 Multi-cloud computing privacy challenges using homomorphic encryption

Multi-cloud has many advantages for the security of user data in cloud computing. Multi-cloud security is one of the concerns that requires a lot of attention. Many researchers and industry professionals debate on security challenges such as isolation management, data exposure and confidentiality, VM security, trust, and special security risks linked to the cooperation between cloud entities. Trust, policy, and privacy, in particular, are key considerations in multi-cloud systems. We will focus on the customer data protection and identity in this thesis. Cloud data privacy is critical because sensitive customer information should not be shared with anyone who is not authorized to access it. When data is stored in many clouds, there should be a method in place to protect data privacy and identification. When the volume of data is extremely sensitive, clients must disguise their identification traits from Cloud computing services in order to maintain anonymity. To protect unwanted access during data transportation and storage in cloud systems, appropriate data encoding techniques should be utilized.

4.1.3 Multi-key Homomorphic encryption

López-Alt et al. [71] proposed the first multi-key homomorphic encryption, a system to provide a homomorphic evaluation on cipher texts encrypted with various keys. Unlike general HE schemes, this kind of HE scheme eliminates the necessity for a key setup step prior to any computation to build a joint key from individual keys. Instead, a cloud evaluator may dynamically convert cipher texts from

encryption using individual keys for encryption using the concatenation of individual users' keys.

We have proposed a solution using the following scheme [72], named MK-TFHE scheme, a first implementation in the literature to implement an MKHE scheme which is defined by seven probabilistic polynomial-time (PPT) algorithms :

— $pp \leftarrow$ MK-TFHE.Setup(1^k) : This algorithm outputs public parameters pp given security parameter k

— $sk \leftarrow$ MK-TFHE.KeyGen(pp) : This algorithm randomly generates secret key sk given the public parameters pp.

— $ct \leftarrow$ MK-TFHE.Encrypt(m, sk) : This randomised algorithm encrypts message m with secret key sk and outputs ciphertext ct=(b, a).

— $(ct^*, T^*) \leftarrow$ MK-TFHE.Pre-process ($\overline{ct}$=(b,a_1,...,a_{ki}),T=id_1,...,id_k) : This algorithm basically extends the input cipher text with additional 0's in order to be able to perform the homomorphic operation over all the underlying k keys.

— $ct^* \leftarrow$ MK-TFHE.Eval(C,ct^*) : This algorithm evaluates circuit C over the l ciphertexts ecrypted with multiple keys.

— $u_i \leftarrow$ MK-TFHE.PartialDecrypt(a_i^*, sk_{id_i}) : This algorithm takes as input a_i^* from ciphertext ct^* corresponding to the party holding secret key sk_{id_i} and outputs a partially decrypted information u_i

— $m' \leftarrow$ MK-TFHE.Merge($b, u_1, ..., u_k$) : This algorithm takes as input all the partial decryptions derived from a ciphertext ct^* and outputs the final plaintext m'.

4.1.4 Our contribution

To further enhance data privacy and reduce the amount of calculations, we proposed more secure system models by adding more clouds.

Many challenges need to be tackled to apply homomorphic encryption in multi-cloud real-world applications. We have identified the multi-key homomorphic encryption [73] or the multiparty extenstions "Threshold setup" of the library OpenFHE [74] for BGV, BFV, and CKKS schemes, to use them in a multi cloud setup.

We introduce and describe a new architecture proposed for multi-cloud electronic medical records as in Fig. 2. This model uses homomorphic encryption algorithms to ensure individual privacy in

network and multi-cloud environments. The advantage of our proposal is that it is based solely on OpenFHE, a reference open-source library in the field of homomorphic encryption, and can be easily configured by neophytes. We performed simple homomorphic operations to prove the feasibility as it is depicted in the next sub-sections.

4.1.5 Experimental evaluation

We make some experiments in order to evaluate the performance and the usability aspects on a variety of applications of multi-key and multiparty extenstions "Threshold" of homomorphic encryption. In the multi-cloud environment, we use OpenFHE [74] for Homomorphic Encryption and DepSky as a multi-cloud platform.

4.1.5.1 OpenFHE : Open-Source Fully Homomorphic Encryption Library

We use OpenFHE[74] to implement the homomorphic encryption scheme, a new open-source FHE software library that integrates a variety of innovative design concepts from previous FHE libraries such as PALISADE, HElib, and HEAAN. OpenFHE supports various FHE schemes and hardware acceleration backends using a standard Hardware Abstraction Layer (HAL). OpenFHE supports both user-friendly and compiler-friendly modes, with the library automatically performing all maintenance operations such as modulus switching, key switching, and bootstrapping. We choose to use a multi-key homomorphic encryption scheme [72], a cryptosystem that allows us to evaluate an arithmetic circuit on ciphertexts, possibly encrypted under different keys.

4.1.5.2 DepSky : Multi cloud computing platform

We experiment with our proposal by using DepSky[75] with local storage, a multi-cloud platform that enhances the integrity, confidentiality, and accessibility of data stored in the cloud. This is accomplished by creating a cloud-of-clouds by encrypting, enclosing, and replicating all the data across a number of separate clouds. This architecture addresses the single cloud's limitations by replicating all of the data in a set, the availability issue of clouds and as a result the data can be retrieved correctly even if some of the clouds corrupt or lost data. It addresses the loss and corruption of data issue by using Byzantine fault-tolerance replication to store data in multi-clouds. It addresses the loss of confidentiality issue by employing a secret sharing schema and erasure codes to ensure that all data

that will be stored in a multi-cloud environment are encrypted.

4.1.5.3 The Health-Care Use-Case

Nowadays, the application of multi-cloud environment is more and more widespread. Multi-cloud environment is applied in many different aspects such as information communication, vehicle systems and medical health systems. While people are paying more and more attention to their medical health, their medical data considered as sensitive must be handled in a secure way like shown in Fig.1.

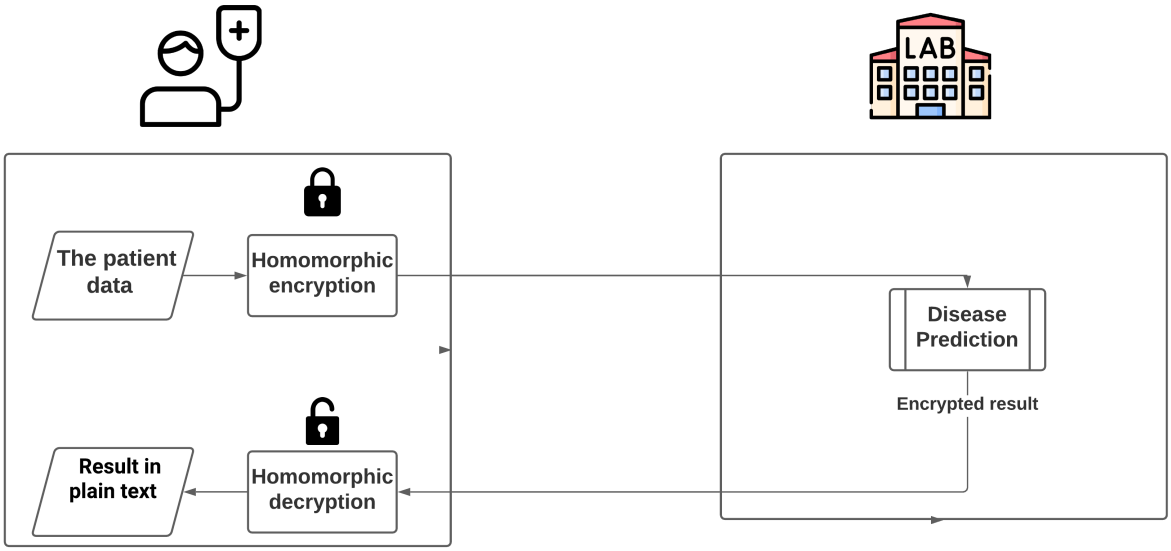


FIGURE 4.1 – "Healthcare monitoring with a single cloud."

Several health-care activities using EHRs (Electronic Health Records) are an attractive use-case for multi-provider cloud in the healthcare industry : data access by the patients, prescription management application for doctors or other institutions, and assisted surgery.

The protection of personal health data is a major issue. Indeed, these data can lead to the covetousness of malicious parties with the aim of generating profits [76]. The three fundamental and main goals of security are : confidentiality, integrity and availability.

— Confidentiality : Only an authorized person should have access to the information.

— Integrity : Information should be correct, and an unauthorized person should not alter it.

— Availability : Information should be accessible, available, and usable at any time but only by an authorized entity.

4.1.5.4 Architecture Model

Because many applications may use sensitive data that are distributed over multiple clouds, in our proposal, we propose a new architecture model that is based on homomorphic encryption in a multi-cloud environment. By combining these two paradigms, we design an architecture model which can ensure the security and the privacy of the users. The proposed architecture for our proposal is summarized in Fig. 2.

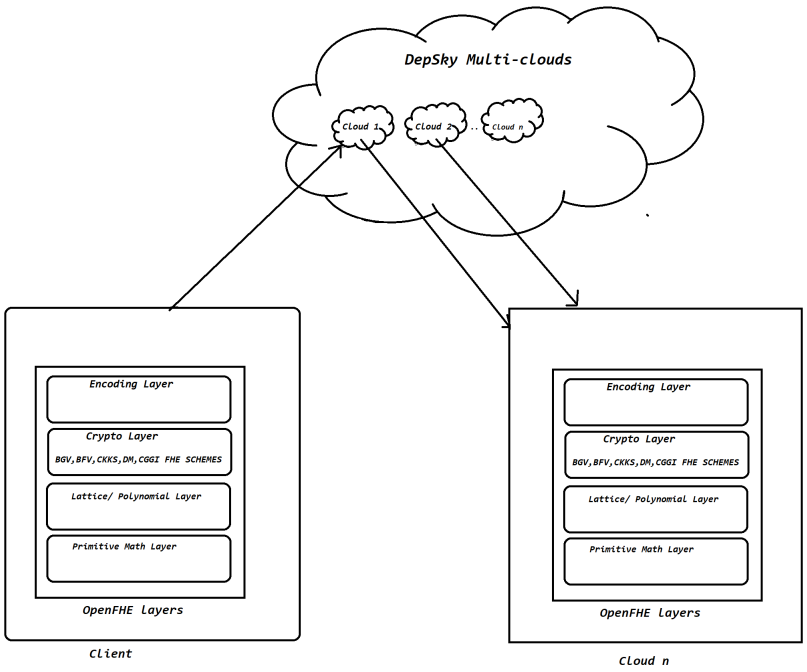


FIGURE 4.2 – "diagram showing how to manipulate encrypted data on a multi-cloud platform DepSky. On the left, the client encrypts the data before sending it to the cloud, on the right, the cloud service can process on it."

4.1.6 Detailed experimental results

Our solution has been implemented using OpenFHE and DepSky, and tested on Linux Ubuntu 64-bit machine with i7-7700 CPU 3.60GHz with four clouds.

The advantage of our architecture is that it is possible to choose among several homomorphic encryption schemes depending on the needs of homomorphic operations. We did our simulation using

four local clouds, nevertheless, it is easy to deploy by choosing a commercial cloud like Amazon S3, Google Storage, RackSpace Files or Windows Azure Storage.

For security parameters, we have used the recommended parameters [72].

Table 1 describes the parameter used for our experimentation.

LWE				RLWE	
n	α	B	d	N	β
560	3.05*10^{-5}	2^2	8	1024	3.72*10^{-9}

TABLE 4.1 – Recommended parameter settings for our MKTFHE scheme : n, α and N, β denote the dimension and the standard deviations for LWE and RLWE ciphertexts to achieve at least 110-bit security level

To show the feasibility of a homomorphic computation on a multi-cloud service we have applied simple operations such as an addition or a multiplication between two integers, the results obtained are presented in Table 2.

The results obtained are promising compared to those obtained with a mono-cloud service, because indeed the multi-cloud architecture allows to make parallel calculations over multiple data encrypted with multiple keys, with an additional negligible cost when we perform the decryption operation using the concatenation of individual user's keys.

It would now be interesting to test more complex operations such as applying machine learning algorithms on health data by using homomorphic encryption [77].

Despite the benefits of cloud applications for healthcare, cloud security challenges must be addressed. In this thesis, we introduce and describe a proposed new architecture for Electronic Health Records with Multi-Clouds. This model will ensure the privacy of people in a network and multi-cloud environment using a multi-key homomorphic encryption algorithm. The advantage of our proposal is that

Number of clouds	**time of one addition**	**Time of a multiplication**	**Blind rotation key**	**Key-switching key**
1	63 μs	1200 μs	0.62 MB	70.3 MB
2	48 μs	580 μs	0.82 MB	79.1 MB
4	25 μs	320 μs	1.03 MB	95.1 MB
8	15 μs	203 μs	1.33 MB	100.3 MB
16	9 μs	124 μs	1.62 MB	120.2 MB

TABLE 4.2 – Results obtained of the calculation time of an addition and a multiplication by varying the number of clouds, with the size of the blind rotation key and the key-switching key

it is based only on an opensource library ”OpenFHE” that is a reference in the field of homomorphic encryption, and the configuration is easy for a non-expert.

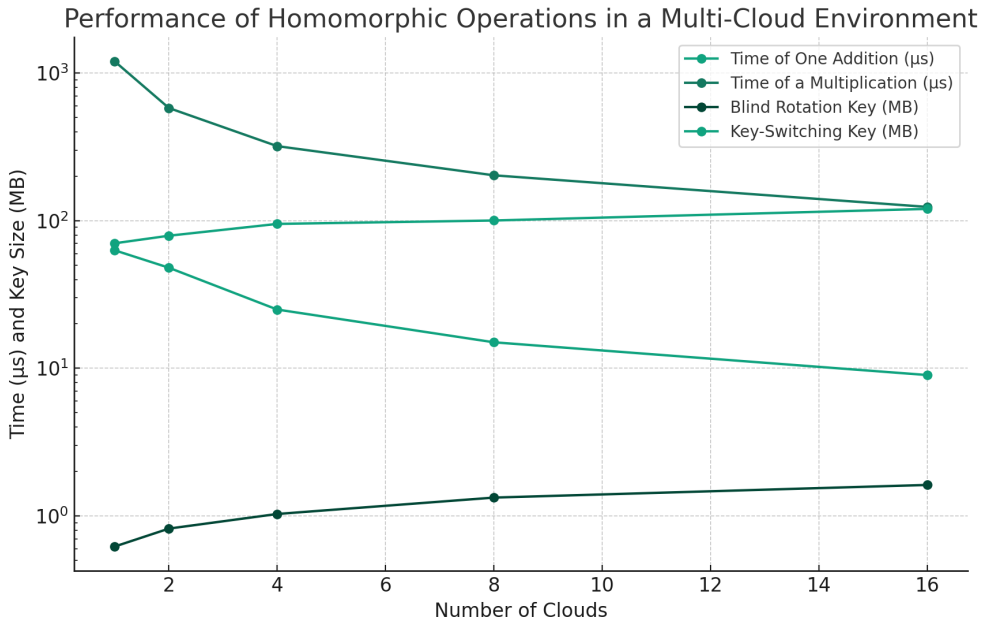


FIGURE 4.3 – ”Performance Analysis of Homomorphic Operations in a Multi-Cloud Environment.”

The figure 4.3 illustrates the performance of homomorphic operations in a multi-cloud environment, as presented in Table 2 of your text. The results clearly show how the time needed to perform an addition or multiplication, as well as the size of the encryption keys (blind rotation key and permutation key) evolve as a function of the number of clouds involved.

The time required to perform an addition or multiplication decreases significantly as the number of clouds increases. This downward trend suggests that multi-cloud architecture enables more efficient parallel computations, thus improving overall performance.

The size of encryption keys also increases with the number of clouds, but this increase is relatively modest compared to the performance gains achieved, underlining the effectiveness of multi-cloud architecture for homomorphic operations.

This graph supports the argument that multi-cloud architecture is promising, particularly for more complex operations such as applying machine learning algorithms to homomorphically encrypted healthcare data.

Chapitre 5

Secure and non-interactive k-NN classifier using symmetric fully homomorphic encryption

5.1 INTRODUCTION

Machine learning as a service" (MLaaS) in the cloud accelerates the adoption of machine learning techniques. Nevertheless, the externalization of data on the cloud raises a serious vulnerability issue because it requires disclosing private data to the cloud provider. This work deals with this problem and brings a solution for the K-nearest neighbors (k-NN) algorithm with a homomorphic encryption scheme (called TFHE) by operating on end-to-end encrypted data while preserving privacy. The proposed solution addresses all stages of k-NN algorithm with fully encrypted data, including the majority vote for the class-label assignment. Unlike existing techniques, our solution does not require intermediate interactions between the server and the client when executing the classification task. Our algorithm has been assessed with quantitative variables and has demonstrated its efficiency on large and relevant real-world datasets while scaling well across different parameters on simulated data.

Cloud services have become central to data storage and data exploitation. Among these services, a machine-learning service is offered to train different models to predict for decision-making purposes. However, this raises the issue of data security because the data processed by the cloud may be sensitive and confidential while belonging to entities that do not trust the cloud provider.

The most commonly used cryptographic techniques to achieve privacy-preserving for machine learning are secret sharing, multi-party computation, and homomorphic encryption (HE). Several tech-

niques ensure strong privacy protection, often at the expense of reduced speed and communication.

One way to solve the computational cost problem inherent to HE is to investigate less complex supervised methods. The k-nearest neighbors (k-NN) approach presents several advantages. Indeed, for a predefined number of neighbors k, the model does not require any training step, the value of the response variable for a given individual is obtained directly from the values observed on the neighbors without needing to estimate new parameters. In addition, the method can handle continuous, categorical, and mixed data. Furthermore, as a non-parametric method, k-NN can be relevant for many data structures as long as the number of observations is sufficiently large.

The prediction for a new observation is obtained by :

— Identifying the k nearest neighbors (according to a given distance)

— Computing the majority class among them (for a classification problem) or by averaging values (for a regression problem).

Unlike other existing works, in this work, we propose a new methodology to apply k-NN on encrypted data by using fully homomorphic encryption avoiding interaction between entities.

We consider a client/service provider architecture that supports scenarios described here. The service provider is a company that owns a labeled training dataset D composed of sensitive data allowing predict for novel observation by k-NN. This model is offered as a service.

We assume a context that is concerned by some privacy issues as in the following :

— Because the training data are sensitive, they cannot be shared with a third party such as a client.

— The model is an intellectual property of the service provider. Hence, the service provider does not want to share the used model with his clients.

— A client who needs to perform classification on the service-provider platform does not trust the service provider.

This work aims to do a classification using k-NN algorithm on sensitive data using HE. Our solution assumes that the service provider, which is the dataset owner, has all the necessary resources to perform the data classification and storage. This assumption ensures that encrypting the training dataset is not necessary since these data are kept with the data owner. Only the client will need to encrypt his query that includes his data by using a private key and by sending it to the data owner

for classification. The goal is to protect the training dataset, the query, and the model parameter k. Our solution meets the following privacy requirement as in the following :

— The contents of D are known only by the data owner since they are not sent to other parties.
— The client's query is not revealed to the data owner.
— The client knows only the predicted class.
— The index of the k nearest neighbors is unknown from the data owner or the client.

Our solution has a greater added value than the existing literature solutions. First, it guarantees that no information leakage occurs during the process : the only things known by the data owner are the dataset and the model used. The only things that the client knows are the query and the class. All intermediate results are encrypted. In addition, our solution is fully non-interactive since prediction is performed by the data owner and do not need any decryption during the process. Finally, it supports multi-label classification.

The rest is organized as follows : Section 4.2.1 presents the background of Functional Bootstrap in TFHE before highlighting the principle of Fast Fully homomorphic encryption over the Torus (TFHE). Our proposed solution is then described in Section 4.3. A simulation study is presented in Section 4.4 to assess our methodology based on real datasets. Finally, Section 4.5 concludes the chapter.

5.2 BACKGROUND

5.2.1 Functional Bootstrap in TFHE

This thesis uses "TFHE : Fast Fully homomorphic encryption over the Torus" as an RLWE-based scheme in his fully homomorphic setting, especially in gate bootstrapping mode. The bootstrapping procedure is the homomorphic evaluation of a decryption circuit on the encryption of a secret key.

TFHE defines three types of ciphertexts, TLWE Sample, TRLWE Sample and TRGSW Sample.

There are two bootstraps algorithms in TFHE. Gate Bootstrap was introduced to implement logic gates, and the Circuit Bootstrap, which converts TLWE samples to TRGSW samples. In our work, we use Functional Bootstrap. By "Functional" we mean that the bootstrap can evaluate functions using a BlindRotate algorithm to perform a lookup table (Lut) evaluation. LUTs are a simple, efficient way of evaluating discretized functions. For instance the sign function was used in [78] and [79].

5.3 OUR CONTRIBUTION

5.3.1 The System Model

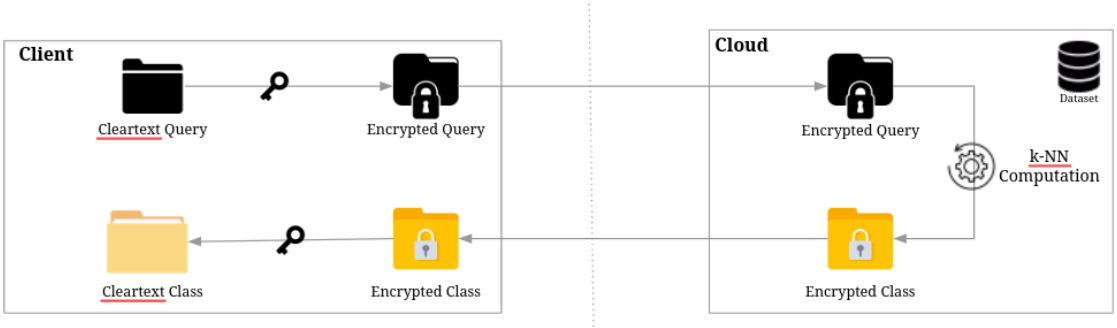


FIGURE 5.1 – The system model : the client is the querier, and the server is the data owner : the data owner receives the query in an encrypted way, performs an encrypted k-NN algorithm then sends the result to the querier for decryption.

Our system uses the client-server architecture (see Figure 1). The client is the querier, and the server is the data owner.

1. The data owner : owns the data and can do heavy calculations. For example, it receives the query in an encrypted way, performs an encrypted k-NN algorithm then sends the result to the querier for decryption.
2. The querier : generates the keys, encrypts the query that contains its data and sends it to the data owner for computations before decrypting the result. The querier can be an ordinary computer or any IoT device that collects data.

5.3.2 Encrypted k-NN Challenges

In order to propose an encrypted version of k-NN, we should substitute the challenging operations used in the standard k-NN with equivalent operations in encrypted domains. As seen before, k-NN is composed of three parts : the distance calculation, the distance sorting, the selection of the k nearest neighbors, and the majority vote.

This subsection will introduce the equivalent operations as integrated with our solution.

5.3.2.1 Distance Calculation

The euclidean distance calculation between the dataset entries x_i as well as the query q are necessary in order to find the k nearest neighbors to the query. We can use the standard formula of the distance as in (1).

$$d^2(x_i, q) = \sum_{j=0}^{p} x_{ij}^2 + \sum_{j=0}^{p} q_j^2 - 2 * \sum_{j=0}^{p} x_{ij} q_j \tag{5.1}$$

What is relevant in our case is the difference between two distances to compare them. So, we get the formula (2) as follows :

$$d^2(x_i, q) - d^2(x_{i'}, q) = \sum_{j=0}^{p} (x_{ij}^2 - x_{i'j}^2) - 2 * \sum_{j=0}^{p} (x_{i'j} - x_{ij}) q_j \tag{5.2}$$

Since the dataset is a clear text, we can easily calculate formula (2) using the TFHE scheme. However, we need to adapt it. Using TFHE, the difference between the distances should be in the range of $[-\frac{1}{2}, \frac{1}{2}]$. Another constraint is that the multiplication is done between a clear-text integer and a ciphertext. Two rescaling values are required to resolve these constraints. Let v be the first one. It is used to have values of the differences between $[-\frac{1}{2}, \frac{1}{2}]$. Let p be the second one. It indicates the precision of the differences. Each attribute of the dataset as well as the query are rescaled using v. p is used when calculating the product $(x_{i'j} - x_{ij})q_j$.

5.3.2.2 Sorting

Sorting computed distances is a crucial step in k-NN. The standard algorithm for sorting, like the bubble sort, can be used while considering encrypted data. However, these algorithms are time-consuming in an encrypted world because the worst case is computed every time. The authors [5] propose two methods to sort an array of values. The method of the direct sort is used in [6]. It is based on a matrix of comparison called delta matrix :

$$\begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,n} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,n} \end{pmatrix}$$

with

$$m_{i,j} = \bar{sign}(X_i - X_j) = \begin{cases} 1 & \text{if } X_i < X_j \\ 0 & \text{else.} \end{cases}$$

When summing columns of this matrix, we will have a sorting index of the distances.

5.3.2.3 Majority Vote

The majority vote can cause a problem because the operation requires comparison to detect the class. To determine the predicted class, we need to know k nearest neighbors' classes. This step is challenging in two ways : first, we need to avoid information leakage, unlike the solutions in the literature. Second, the majority vote requires comparison in order to predict the class.

To the best of our knowledge, no solution in the literature studied this point in an encrypted way without information leakage. Therefore, in the following subsection, we will demonstrate a solution to process k-NN with the majority vote in an encrypted way while supporting a multi-label classification.

5.3.3 Our proposed k-NN algorithm

Our proposed algorithm, called "HE-kNN-V", is composed of three steps : the construction of the delta matrix, the selection of k-nearest neighbors, and the majority vote. The two first steps are similar to the solution of [6] even if we adapt the existing formulas in order to eliminate unnecessary calculations. [6] use polynomials to define the formulas, while what interests us is just one term of those polynomials to eliminate non necessary calculations. The majority vote is our added value and is specific to our solution. We will discuss in this subsection the design of our solution, including each building block.

5.3.3.1 Building the delta matrix

To build the delta matrix, we need to know the sign of the differences between the distances to sort. Since we defined a method to calculate the differences in the last subsection, the sign can easily be achieved using the standard bootstrapping sign function in TFHE. However, the standard bootstrapping function returns +1 if the phase is greater than 0 and -1 if the phase is lower than 0. Therefore, since we need to have 0 or 1 in the matrix, we need to adapt the bootstrapping operation to return $\frac{1}{2}$ and $-\frac{1}{2}$ then by adding $\frac{1}{2}$ to the result we will have 0 or 1.

Even if building this matrix is time-consuming, it is highly parallelizable.

5.3.3.2 Selecting the k-nearest neighbors

To select the k–nearest neighbors, we use the scoring operation proposed by Zuber [6]. By using the delta matrix, the principle is as follows :

1. Sum m values in each column with m the number of possible operations without bootstrapping.
2. If there are still values to sum : do a bootstrapping operation using the modified sign bootstrapping function (See Algorithm 1 in [6]) and go to Step 1.
3. Otherwise, execute the modified sign bootstrapping and return the last sign returned by this operation.

Finally, we obtain an encrypted vector where the position i equals the cipher of 1 if the individual with index i is among the k-nearest neighbors, the cipher of 0 otherwise. We call this vector the "mask" (See Figure 2 for more clarity).

5.3.3.3 Majority vote

The majority vote is the most important added value in our work. We propose to do the majority vote without any leakage of information, unlike existing works like that of [6] in which the majority value is done in clear text or by using other alternative solutions proposed in the literature.

First, we illustrate the issue with the method of [6]. We consider the scenario where the querier does the calculations. The majority vote is done in clear text, but we need to decrypt the vector of indexes of the nearest neighbor. The data owner does the decryption. Significant information leakage occurs if the data owner knows the vector of indexes. Then, he will know the classification of the query, and by doing some triangulation, he can approximate the query. In addition, the solution will be interactive. If we consider the scenario where the data owner does the calculation, the decryption of the vector is done by the querier. However, to do the classification, the querier should know the labels of the dataset, which is also a critical leakage of information. In addition, the querier will know the size of the dataset and the k parameter of nearest neighbors considered. This information is considered as internal information of the model used, and it should be protected.

In our solution, the majority vote is done by the data owner in an encrypted way. First, the data owner encodes the labels using one hot encoding. Having the mask and the matrix of labels in one hot form, it is easy to do an AND operation between the mask and each column of the labels, as in

Figure 2. We get a matrix A (for affectation) with A_{ij} equal to 1 if the individual i is among the k-nearest neighbors and its class is j. Using this matrix, it is possible to sum the columns and obtain the probability of each class. We can now return only the class and guarantee no information leakage and no interactivity.

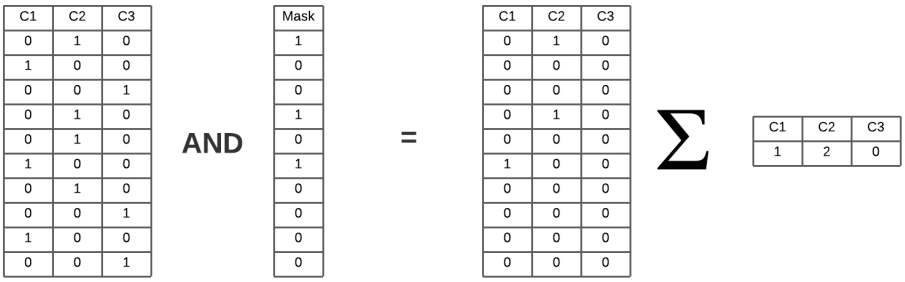


FIGURE 5.2 – Majority Vote illustration by using the mask

5.4 PERFORMANCE EVALUATION

In this section, we discuss the experiments of our solution. First, we describe the technical and the setup of the environment. Then, we will evaluate the performances of our solution according to different criteria : execution time, accuracy, bandwidth consumption.

5.4.1 Test Environment

5.4.1.1 Setup

Our solution is implemented using the TFHE scheme in C/C++ and Python for training k-NN in clear text and for tests. To test the effect of parallelism, we used OpenMP to do some parallelization. The source code is available in the following github "https ://github.com/Yulliwas/HE-kNN-V". Our solution is tested on Linux Ubuntu 64-bit machine with i7-8700 CPU 3.20GHz.

Table 1 shows the parameters used to setup TFHE scheme.

5.4.1.2 Datasets

To test our solution, we choose to use 6 datasets : Iris, Breast Cancer, Wine, Heart, Glass and MNIST as in Table 3. The goal is to test the performances of our algorithm in different distributions

λ	**N**	σ
110	1024	10^{-9}

TABLE 5.1 – TFHE Parameters :
λ for the overall security, N for the size of the polynomials,σ for the Gaussian noise parameter.

m	v	p	b
64	4	1000	4*m-4

TABLE 5.2 – HE-kNN Parameters :
the number of operations m without needing a bootstrapping, the bootstrapping base b, and the rescaling factors v and p

of data, so that to confirm that our solution works with any dataset and that has performances that are equivalent to those of clear-text domains.

Dataset	**n**	**d**	**classes**
Iris	150	4	3
Wine	178	13	3
Heart	303	75	5
Breast Cancer	699	10	2
Glass	214	10	2
MNIST	1797	10	3

TABLE 5.3 – Datasets :
number of individuals(n), the size of the model (d) and number of classes

5.4.1.3 Simulation procedure

First, we preprocess the data by rescaling each attributes to a value between 0 and 1. Our dataset and the query should be rescaled by a factor of v as seen above. We must also multiply the dataset vectors by the precision factor τ and then rounded. In the other hand, the query vector is divided by this same factor. To obtain the classification rate, first we need to divide our dataset to a training set and a test set. We choose to use 20% of our dataset as a test set and the rest as a training set. Among the training set, we select a certain number of points that represent as well as possible our dataset. The process for choosing the best points that represent our training set is as follows :

1. choose n individuals randomly ;
2. calculate the classification rate ;
3. Repeat the previous Step 1 and Step 2 a certain amount of time and keep the best accuracy and the best individuals.

To select the k parameter, we use the same procedure as in the clear domain. In our case, we tested

different values of k and we keep the best k value that gives the best results.

5.4.2 Performance results

To position our approach according to existing works, and especially regarding the voting step that is performed without information leakage, we compare in Table 4 our solution with Zuber's solution and with a clear-text version based on the Iris dataset and a fixed k=3. The comparison is done in terms of complexity (C), Information Leakage (L), accuracy (A), interactivity (I) and execution time (T). The accuracy and the prediction time are indicated only when it is possible.

Work	**C**	**L**	**I**	**A**	T
HE-kNN-V	$O(n^2)$	N	N	0.97	1.72s
HE-kNN-VP	$O(n^2)$	N	N	0.97	0.46s
Zuber	$O(n^2)$	Y	Y	0.98	1.74s
Clear k-NN	$O(n)$	Y	N	0.95	1.8ms

TABLE 5.4 – Comparison between solutions for Iris Dataset :
complexity (C), Information Leakage (L), accuracy (A), interactivity (I) and execution time (T).

5.4.2.1 Empirical study

5.4.2.1.1 Classification rate To evaluate the classification rate, we have chosen the accuracy instead of other metrics like : recall or F1-score. We studied the accuracy according to two parameters : the number of data sampled from the dataset and the number k of neighbors. The goal is to choose the best points that represent the datasets and the best k parameters for each dataset.

We chose real-world datasets in order to see the evolution of the accuracy and compared it to clear-text accuracy.

In one hand, we know that the accuracy depends on the k parameter and we can confirm it easily in the graphs. On the other hand, the assumption that the accuracy depends on the number of data used is not complete. For the dataset where the data is well separated (like Iris), having a lot of data is not necessary, the best accuracy can be achieved using only few data. But, in the case where data is not well separated (like in Heart dataset), the accuracy seems to depend on the number of data.

According to our different simulations illustrated in Figure 3 and Figure 4, we do not lose accuracy when we apply our HE-kNN-V method on the encrypted data compared to the application of the kNN on the plain data. This is possible by varying the number of individuals and by fixing k to 3.

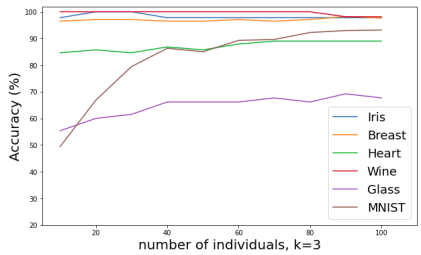


FIGURE 5.3 – Encrypted Accuracy vs number of individuals

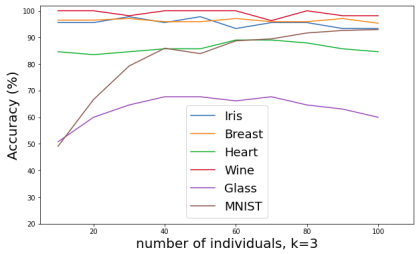


FIGURE 5.4 – Clear-text Accuracy vs number of attributes

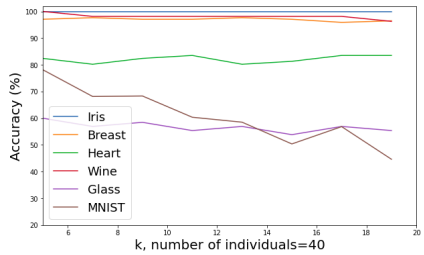


FIGURE 5.5 – Encrypted Accuracy vs k-parameter

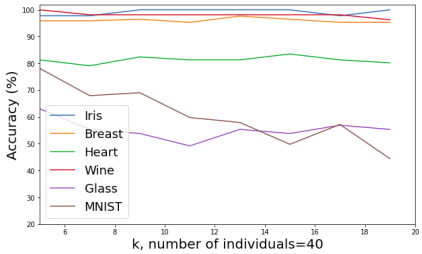


FIGURE 5.6 – Clear-text Accuracy vs k-parameter

We also notice that by setting the number of individuals to 40 and varying k, (see Figure 5 and Figure 6) the accuracy behaves in the same way between the application of the kNN on the plain data and the application of our method HE-kNN-V on the encrypted data.

5.4.2.1.2 Execution time In our solution, the execution time is independent of the content of the dataset, it does not depend on the values, but does depend on the content, since it depends on the number of tuples . We can use either simulated dataset or real world dataset. To visualize the evolution of the execution time according to k, n and d, we choose to use the Breast Cancer dataset instead of simulating a new dataset. We change n, k, d and we see the evolution of the execution time.

Our simulations, as depicted in Figure 7, illustrate that HE-kNN-V is parallelizable, and also that the number of individuals strongly impacts the execution time unlike the two simulations of Figure 8 and Figure 9 where the variation of respectively d the number of attributes and k does not impact the execution time.

5.4.2.1.3 Bandwidth In our solution, the only thing that is communicated is the query in the ciphertext and the response in the ciphertext. The size of the query is proportional to the number of

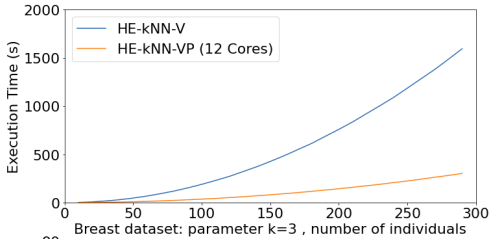


FIGURE 5.7 – Execution time vs number of individuals

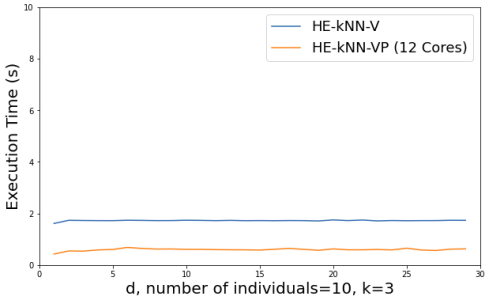


FIGURE 5.8 – Execution time vs number of attributes

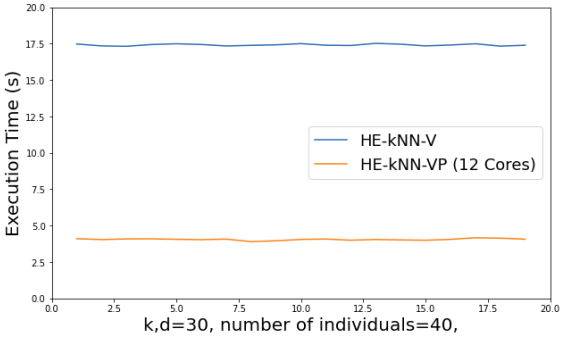


FIGURE 5.9 – Execution time vs k-parameter

attributes d. Each attribute is a TLWE Sample with the size of 4 KB and the size of the response (number of classes)*4 KB. The bandwidth according to each dataset is illustrated in Table 5.

TABLE 5.5 – Bandwidth

Dataset	**Bandwidth (KB)**
Iris	28
Wine	64
Heart	64
Breast Cancer	128
Glass	60
MNIST	296

5.4.2.1.4 Discussion According to our experiments, we can say that the accuracy in our case depends on three factors : the number of individuals, the representativity of these individuals and the k parameter. To have a better model that fits our dataset, we must select the individuals that are more representative of our dataset and the best k parameter. We also should take care of the number of individuals because most of the execution time depends on that number.

5.5 CONCLUSION

We proposed HE-kNN-V a method for performing k-NN on encrypted data that includes a majority vote for class-label assignment. The proposed solution addresses all stages of k-NN algorithm with fully encrypted data. It guarantees that no information leakage occurs during the process. Unlike other techniques, our solution eliminates the need for intermediate interactions between the server and the client when performing classification tasks. Our algorithm has been evaluated using quantitative variables and demonstrated its efficiency on large and relevant real-world data sets. As a perspective, it would be interesting to see how a hardware acceleration of the TFHE scheme could improve the computation time of our proposed solution HE-kNN-V.

Chapitre 6

Secure k-means clustering using TFHE

6.1 INTRODUCTION

Cloud computing offers a solution to the lack of computational and storage resources in various domains, allowing users to outsource high-cost calculations. Machine Learning (ML), being resource-intensive, benefits from cloud ML platforms provided by IT companies. This "Machine Learning as a Service" (MLaaS) allows users to utilize cloud resources for ML tasks.

Despite its advantages, sectors like healthcare, finance, and government hesitate to use cloud resources due to privacy and security concerns, as data in the cloud is often processed in clear text. To address this, solutions like anonymization, cryptography, and perturbation are used. Anonymization changes data to prevent recognition by attackers, while perturbation adds noise to randomize data. However, these methods have limitations in fully securing data.

One effective cryptographic solution is Homomorphic Encryption (HE), allowing arithmetic operations on encrypted data without decryption. Introduced in 1978, HE comes in three forms : Partial Homomorphic Encryption (PHE) for single-operation tasks, Somewhat Homomorphic Encryption (SHE) for limited operations, and Fully Homomorphic Encryption (FHE) for unlimited operations using addition and multiplication. In this work, we aim to achieve a privacy-preservation clustering using FHE. Specifically, we are interested in doing a k-means clustering by proposing a new secure and efficient outsourced clustering. While k-means is widely used, simple and easy to understand, implementing its secure version using HE is not trivial because k-means requires more complex operations such as division and comparison in addition to addition and multiplication. We can decide to do these operations locally in clear text by using intermediate decryption, but the purpose of using cloud

computing is foremost to limit calculations in the user side. Indeed, according to the state of the art of HE, it is not possible to do all the operations in the cloud server easily, such as the equality/inequality test. Number comparison and sign determination are critical processes in a homomorphic setup.

Therefore, the solution proposed in this work aims to limit interaction between the cloud server and the user/client using a specific type of FHE called TFHE (Fast Fully Homomorphic Encryption over the Torus). Using this specific scheme, we can benefit from a functionality offered by TFHE to do comparisons in an encrypted way. Therefore, we limit the interaction to the sole calculation of the new centroids of the clusters.

The remainder of this chapter is organized as follows : Section 5.2 presents a review of the related work. Section 5.3 provides background about k-means algorithm. Section 5.4 details the proposed solution for k-means using TFHE before highlighting the experiments that are undertaken to validate the proposed algorithm in Section 5.5, and before concluding in Section 5.6.

6.2 Related works

Many research works have used HE to secure k-means according to one of the two settings : distributed clustering or individual clustering. All these works have attempted to find alternatives to perform distances calculation, comparisons, and divisions securely.

In the distributed-clustering context, many actors have different datasets and collaborate jointly to have a common clustering.

Liu et al., in [80], proposed a solution for a two-entities based k-means clustering. In this solution, the authors use two schemes : Liu scheme [**liu**] and Pallier scheme [9]. Each entity encrypts its data and sends them to the cloud. In the cloud, they use trapdoor information given by the data owners to do the comparisons, then the data owners collaborate to compute the new centroids.

Jiang et al., in [81], used the Pallier additive scheme and proposed a k-means clustering where an intermediate decryption is necessary to compute the euclidean distances and opted for a bit-a-bit comparison which is time consuming.

In the individual clustering, one entity owns the dataset and then outsources it to the cloud for k-means clustering purposes.

Theodouli et al., in [82], used the BV (Brakerski-Vaikunathan) scheme to design a solution that allows collaboration between a server and a client. In this work, the authors studied three scenarios. In the first scenario, only the calculation of the distance is performed within the cloud. The rest of the calculations is done in the client side. In the second scenario, only the comparison and the division are done in the client side. However, this solution leads to information leakage regarding the assignment of the individuals. To solve this problem, the third scenario used an affectation vector instead of passing the class directly.

Almutairi et al., in [83], propose k-means method that limits the interaction with the data owner using the concept of an "updatable distance matrix (UDM)". The latter is a 3D matrix where the first two dimensions correspond to the number of data in the dataset and the third to the number of attributes. Each cell of the matrix is initialized with different attributes of the data vector. The idea is to store encrypted data and his UDM matrix in a third party. This matrix is updated every k-means iteration using the offset matrix obtained by computing the difference between the new center and the current center. This method consumes time and memory to store the UDM matrix.

Jäschke and Armkne, in [84], used the TFHE scheme to implement a secure k-means at binary level. This is possible by designing a binary circuit. Theoretically, this method gives the same results as the clear version, but it remains impractical due to the execution time which is astronomical even when executed in a cloud server.

Sakellariou and Gounaris, in [85], introduced a third trusted entity to decrypt the intermediate results using an equivalent key (but different) to the key used to encrypt data. They used the BGV (Brakerski, Gentry and Vaikuntanathan) scheme, and considered the honnest-but-auditable threat model. This means that we assure that the trusted entity does only the decryption of the intermediate results. If the third entity doesn't follow the protocol for which it was designed, an anomaly is sent to the data owner.

6.3 BACKGROUND

6.3.1 k-means algorithm

In this section, we provide an overview of the k-means clustering algorithm. In our contribution, this algorithm will be adapted to be executed in a secure way using homomorphic encryption.

Given a dataset, k-means algorithm partitions individuals in k subsets called clusters. Each cluster is represented by its center (called also centroid), and each individual belongs to the cluster that has the shortest euclidean distance (or squared distance) between its centroid and this individual. The optimization problem is defined as follows :

$$\min_{c_1,...,c_K,C_1,...,C_K} \sum_{k=1}^{K} \sum_{i \in C_k} (x_i - c_k)^2$$

The algorithm works as follows : First, we select k initial cluster centers. Using Forgy initialization allows to select the centroids randomly. We execute then iteratively the clustering according to two steps : the assignment step and the center-updating step. In the assignment step, we affect all the observations to the closest cluster using the euclidean distances between the observations and the centroids. In the updating step, we compute the new centroids by averaging each attribute over all the objects allocated into the same clusters. These centers are now the new centroids to use in the next iteration. The algorithm terminates when the centroids do not change any more.

6.3.2 Clustering evaluation

After executing a clustering algorithm, we need to evaluate the results of our algorithm. This evaluation depends on our goals and we need to take care of different aspects : we need to detect the exact number of clusters, but also to evaluate the quality of the clustering without adding any external information. There are two types of evaluation : internal evaluation and external evaluation as described in the next sub-section.

6.3.2.1 Internal evaluation

In the internal evaluation, we evaluate the results without using external information. It is based on the compactness and separation between clusters. A good clustering algorithm maximizes the

compactness inside a cluster and the separation between clusters. We describe in the following two metrics : the inertia and the silhouette coefficient since we will use them in our proposal.

— **Inertia :** also called within-cluster sum-of-squares, it measures how well a dataset was clustered by k-means algorithm. It is calculated by measuring the distance between each data point and its centroid, squaring this distance, and summing these squares across one cluster. Generally, a good clustering maximize separation between clusters and maximize compactness inside clusters i.e we need to have low inertia.

— **Silhouette coefficient :** is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1. Positive values mean that the clusters are distinguished from each other, whereas negative values mean that clusters are assigned in a wrong way.

6.3.2.2 External evaluation

The external evaluation proceeds by using external information in order to compare the results of clustering with these information. It is useful to compare clustering results to real labels, but also to compare the results of two clusterings. In our proposal, we use the adjusted rand score and the normalized mutual information as external metrics.

— **Adjusted Rand Score (ARI) :** is introduced to determine whether two cluster results are similar to each other by considering all pairs of individuals and counting pairs that are assigned in the same or in different clusters in the predicted and the true clusterings.

— **Normalized mutual information (NMI) :** is a normalization of the Mutual Information (MI) score to scale the results between 0 (no mutual information) and 1 (perfect correlation), (NMI) which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency.

6.4 OUR CONTRIBUTION

6.4.1 The System Model

As stated in the related work, the existing solutions use the communication and the collaboration between the client and the server to cluster the data. In our work, our goal is to limit the interaction between the client and the server. Hence, as shown in Figure **??**, the maximum of calculation is done

within the server side. As the assignment step is a time-consuming operation, we delegate this step completely to the server side by using the euclidean distance unlike other existing works. By this way, the client will just encrypt the dataset, send it to the cloud server to perform the assignment step and return the results to the client that computes the new centroids and tests the termination condition. Since the division and the comparison operations are not possible using HE, we opt to a functional bootstrapping to do comparisons in an encrypted way.

In the next section, we will explain the building blocks in a clear setting then we will adapt them to an FHE setting.

6.4.2 In clear setting

Before executing k-means algorithm, we do a Z-score normalization and a MinMax Standardization. The Z-score normalization is useful to have the same weight for all the attributes. As we use TFHE and our data should be in the torus, we use Min Max Normalization to have our data in the [0,1[interval.

6.4.2.1 Initialization

In order to fix the initial centers to start the algorithm, there are many methods to initialize the algorithm in the literature : Picking the centers manually, Forgy initialization, k-means++,etc. In our solution, we choose to use the Forgy initialization, by picking the initial centroid randomly.

Since two different initialization may lead to different clustering and in the purpose of comparing the results of the k-means version implemented in scikit-learn with our solution, we need to have the same initial centers. Therefore, we set the centers manually for the scikit-learn version according to the random initial centers picked in our version.

It is interesting to use k-means++ to initialize the centers but we think that proposing an HE version of k-means++ need to be studied separately.

6.4.2.2 Assignment step

6.4.2.2.1 Distance Calculation To ensure the convergence of k-means, we use the euclidean distance. We compute the distance between an observation a and the centroid C_i. We use the standard formula

of the euclidean distance as in formula 6.1.

$$d_{ai}^2 = \sum_{j=0}^{d} C_{ij}^2 + \sum_{j=0}^{d} a_j^2 - 2 * \sum_{j=0}^{d} C_{ij} a_j \tag{6.1}$$

Using this formula, we need to pass the norm of each observation as a trapdoor information to the server. However, what is relevant is the difference between distances. Hence, we compute only the difference between the squared distances, and we get the formula 6.2. By this way, we do not need to pass any trapdoor information.

$$d_{ai}^2 - d_{ai'}^2 = \sum_{j=0}^{d} (C_{ij}^2 - C_{i'j}^2) + -2 * \sum_{j=0}^{d} (C_{i'j}^2 - C_{ij}^2) a_j \tag{6.2}$$

Since, in our case, we have to do only multiplications by a scalar using the TLWE Sample, we decide to let the centroids in clear setting. We are aware that this can lead to an information leakage. However, to overcome this issue, we can use other techniques like differential privacy. In our solution, we assume that this leakage is not a problem since our goal is to protect data and the result of clustering.

6.4.2.2.2 Delta matrix construction The delta matrix is $K \times k$ matrix that contains the sign of the distance differences as computed in the previous step. This method is used in [6] to find the k-nearest neighbors of an observation. In our case, we use it to find the nearest centroid.

$$\begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,n} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,n} \end{pmatrix}$$

with

$$m_{i,j} = \bar{sign}(X_i - X_j) = \begin{cases} 1 & \text{if } X_i < X_j \\ 0 & \text{else.} \end{cases}$$

When summing columns of this matrix, we obtain a sorting index of the distances between observations and the K centroïds.

6.4.2.2.3 Assignment Vector We can sort distances between $a???$ and the centroids by summing the columns of Δ as in Formula 6.3.

$$(\Delta_0, \Delta_1, ..., \Delta_{k-1}) \text{ with } \Delta_j = \sum_{i=0}^{k-1} \delta_{i,j} \tag{6.3}$$

The closest centroid to a is where $\Delta_i = 0$. If we want to find an assignment vector from the previous index (containing 1 in the position corresponding to the closest centroïd and 0 otherwise), we can apply the function $\overline{sign}$ as in Formula 6.4.

$$\overline{sign}(\Delta_i) = \begin{cases} 1 & \text{if } \Delta_i \leq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6.4}$$

Figure 6.1 illustrates the process of assignment.

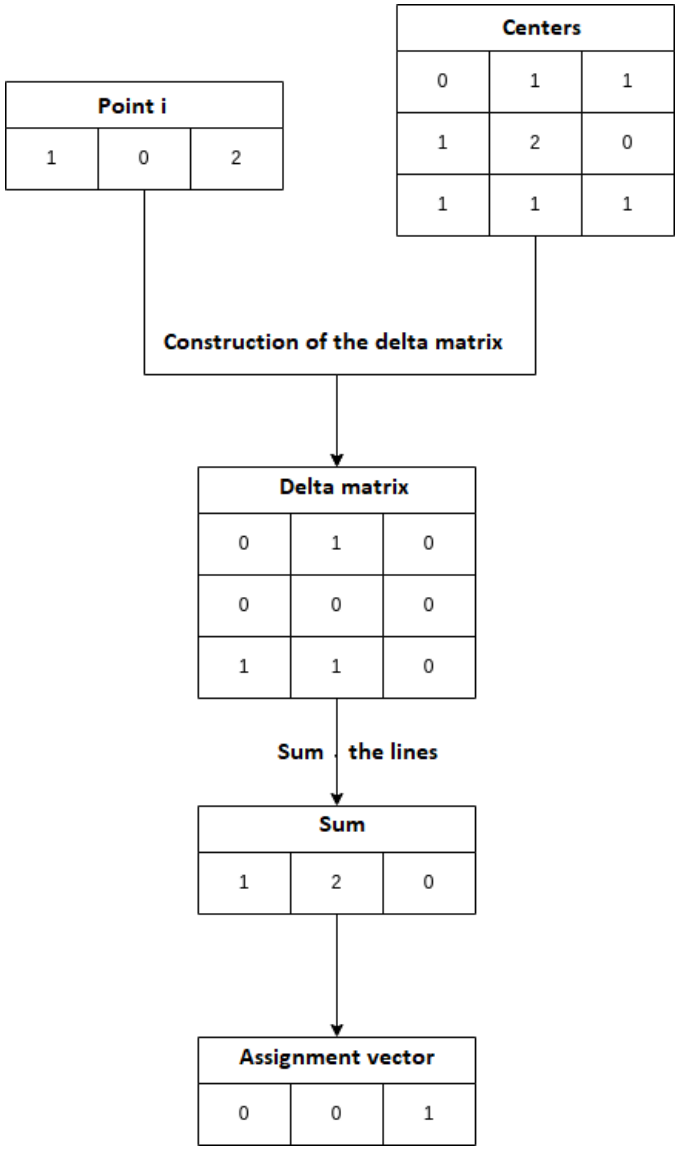


FIGURE 6.1 – Single assignment example

If the number of the centroïds is less than m possible operations without doing bootstrapping, the

solution will work without any problem. In an FHE context, the parameter m can be less than k and the solution is limited. After m additions, we need to reduce the noise accumulated. This is possible by a sum property as in Formula 6.5.

$$\Delta_j = \sum_{i=0}^{k-1} \delta_{i,j} = \sum_{i=0}^{m-1} \delta_{i,j} + \sum_{i=m}^{k-1} \delta_{i,j} \tag{6.5}$$

Thanks to this formula, we can perform a programmable bootstrapping after each m operations. However, it remains difficult to apply this property due to the limitations related to TFHE. Indeed, since TFHE is designed to work with logical circuits, the intermediate result must be in the torus domain. Here, the most relevant part is not the sum but the assignment vector. For this, we use the operation S_g defined by [6] as given in Formula 6.6.

$$S_g(x_1, x_2, ..., x_n) = \max(0, \sum_{l=1}^{n} x_l - g + 1) \text{ with } \sum_{l=1}^{n} x_l \leq g \tag{6.6}$$

This function returns 1 if the centroid i is the closest center to the vector and 0 otherwise. g is the number of operations achievable without bootstrapping.

Example : for $g = 7$,

$S_7(0, 1, 1, 1, 0, 0, 1) = \max(0, 4 - 7 + 1) = 0$

$S_7(0, 1, 1, 1, 1, 1, 1) = \max(0, 6 - 7 + 1) = 0$

$S_7(1, 1, 1, 1, 1, 1, 1) = \max(0, 7 - 7 + 1) = 1$

In our case, we should calculate

$$S_k(\delta_{0,i}, ..., \delta_{i-1,i}, \delta_{i+1,i}, ..., \delta_{k-1,i}) \tag{6.7}$$

In an FHE context and if $k > g$, it is impossible to compute the function sc directly. However, [6] has demonstrated the proposition 1.

Proposition 1 :

Let $x_1, ..., x_{2g-k} \in \{0, 1\}$. We denote

$$A = S_{1,g}(x_1, ..., x_{g-1}, S_{1,g}(x_g, ..., x_{2g-1}))$$

and

$$B = S_{1,2g-1}(x_1, ..., x_{2g-1})$$

In an FHE context, we cannot compute B, because it needs to sum over than g binary values. However, we can compute A and $A = B$.

Based on this proposition, we can compute (6.7). However, we need to take care of the latest bloc S. If the latest bloc doesn't have $g - 1$ elements, it is necessary to add a padding of 1 until the $g - 1$ element.

6.4.2.3 Updating centroids

To update centroids, we use the division operation. In FHE settings, doing this operation is possible at a binary level, but it is time consuming. Since we only need one division in each iteration, we decide to do this operation in a clear way. We are in front of two possibilities : doing the sum in the server side and only the division in the client side, or computing the centroid of the cluster in the client side. We studied the security and the practicality of the two settings.

The first option limits the computations done in the client side, but by doing this the server must know the assignment results. We discarded this option because it represents a flagrant breach of security. We have chosen then the second option even if it generates more calculations in the client side, especially because this option is more appropriate regarding security considerations. In fact, performing additions in a clear text are not resource consuming.

6.4.3 Encrypted k-means with FHE

In the previous subsection, we presented our algorithm in clear setting. In this subsection, we will see how this version can be easily translated to the FHE setting.

6.4.3.1 Encoding and Encrypting

We need to encode our data in a torus. In other words, the values must be between 0 and 1. This can be done in the pre-processing step. We need then to define two bases : one for the delta matrix, we denote it b_δ and the second for the assignment matrix, we denote it b_f. This means that we will not have a matrix of 0 and 1, but a matrix of 0 and $\frac{1}{b}$. Thanks to these bases, we ensure that the inputs and the outputs will be inside the torus.

6.4.3.2 The difference of the squared distances

When computing the distances, the distances must be in the interval $[-\frac{1}{2}, \frac{1}{2}]$ to stay in the torus, and we need to differentiate the negative and the positive values. This is why we need to use a scaling factor v (with $v \geq ||x_i||$).

Since TFHE is limited to scalar multiplications, we propose to keep the centroids in clear text. In one side, we are interested in protecting the dataset as well as the assignment results. In the other side, we need to have more precision of the scalars that represent the centroids. Hence, we introduce a precision factor denoted τ and for each attribute in the centers, we will have the next value as follows : $Round(\frac{\tau \times c_{ij}}{v})$.

We divide all the data vector by τ. The attributes of each vector will have the next value : $\frac{x_{ij}}{\tau v}$.

The euclidean distance is then defined by a scalar product that is formulated as in Formula 6.8.

$$\sum_{j=0}^{d} Round(\frac{\tau \times c_{ij}}{v}) \times \frac{x_{ij}}{\tau v} \tag{6.8}$$

If $[X^i]$ is the ciphertext of the vector to be affected, C^i the centroid in clear text and $||C_k||^2$ is its squared euclidean norm, then the difference is computed using Formula 6.9.

$$2[X^i](C^j - C^k) + ||C_k||^2 - ||C_j||^2 \tag{6.9}$$

For big values of τ, this formula is equivalent to $\frac{1}{v^2}(d_k^2 - d_j^2)$. As we are only interested in the sign of the difference, it is clear that it is the same as the sign of $(d_k^2 - d_j^2)$.

6.4.3.3 Delta matrix

To build the delta matrix, we must know the sign of the differences between the distances to sort. Since we defined a method to compute the differences in the last subsection, the sign can easily be achieved using the standard bootstrapping function in TFHE.

The bootstrapping algorithm defined in [19] returns $\frac{1}{b_\delta}$ if $m > 0$ and $-\frac{1}{b_\delta}$ otherwise. We noticed that this behavior is similar to a *sign* function. However, in our case we must return $\frac{1}{b_\delta}$ if $m > 0$ and 0 otherwise. So we modified the algorithm as in the following :

First, we brought a modification in the line 5 by multiplying the bootstrapping base b_δ by 2. This

Algorithm 4 Sign Operation

Require: a TLWE Sample $(a, b) = [m]$, bk a bootstrapping key, b_δ a bootstrapping base
Ensure: $LWE(\frac{1}{(b_\delta)})$ if $m + \frac{1}{b_\delta} \in [0, \frac{1}{2}]; 0$ else

1: Let $\bar{b} = \lfloor 2Nb \rceil$
2: **for** $i = 1$ to n **do**
3: $\quad \bar{a_i} = \lfloor 2Na_i \rceil$
4: **end for**
5: Let $testv = (1 + X + \cdots + X^{N-1} \times X^{-\frac{2N}{4}} . \frac{1}{2 \times b_\delta})$
6: ACC $\leftarrow [\, X^{\bar{b}}, (0, testv)]$
7: **for** $i = 1$ to n **do**
8: $\quad$ ACC $\leftarrow [\, h + (X^{\bar{a_i}} - 1).bk_i)]\ .ACC$
9: **end for**
10: **return** $SampleExtract(ACC) + [\frac{1}{2 \times b_\delta}]$

modification allows to have as output $\frac{1}{(2 \times b_\delta)}$ and $-\frac{1}{(2 \times b_\delta)}$. This is still not the result needed, but we need to add $\frac{1}{(2 \times b_\delta)}$ to the result in cipher way to have 0 or $\frac{1}{b_\delta}$.

Let's note that building the matrix delta can be totally parallelized. In this case, it is not necessary to compute all the elements of the matrix since $\delta_{i,j} = 1 - \delta_{j,i}$.

6.4.3.4 Affectation vector

The process of computing the affectation vector is not different from the process described with the clear setting. To select the closest centroid, we proceed as follows :

1. Sum m values in each column with m the number of possible operations without bootstrapping.
2. If there are still values to sum : do a bootstrapping operation using the modified sign bootstrapping function (See Algorithm 1) and go to Step 1.
3. Otherwise, execute the sign bootstrapping and return the last sign returned by this operation.

Finally, we obtain an encrypted vector where the position i is equal to the cipher of 1 if the centroid i is the closest one to our vector.

6.5 PERFORMANCE EVALUATION

In this section, we will experiment our algorithm using different datasets. First, we describe the setup of the environment. Then, we will discuss the choice of security parameters and other parameters used in our solution. We evaluate the performances of our algorithm according to different criteria :

adjusted rand index, silhouette coefficient and execution time. We will also discuss our proposal from a security point of view.

6.5.1 Test Environment

Our solution is implemented using the TFHE scheme in C/C++ using the TFHE library [19]. We use Python language for pre-processing and performance-evaluation purposes to benefit from the function implemented in the scikit learn. Our solution is tested on Linux Ubuntu 64-bit machine with i5-1135G7 @ 2.40GHz × 8.

6.5.1.1 Datasets

To test our solution, we have chosen the datasets described in Table 6.1. The variety of datasets will contribute to have a better vision of the performance evolution of our solution.

Datasets	n	d	k
Iris	150	4	3
Wine	178	13	3
Glass	214	9	6
Ovarian	216	100	2
Breast Cancer	699	9	2
MiceProtein	1077	77	8
PenDigits	10992	16	10

TABLE 6.1 – Datasets

6.5.2 TFHE Tests

Our goal in this subsection is to explain our choices concerning the dimension of the LWE problem. We know that it is recommended to have a security parameter greater than 128 bits. To select our parameters, the first step was to look at the literature and to test these parameters using *lwe_estimator* to test the security provided by these parameters (see Table 6.2).

The parameters provided by [19] ensure the minimal security provided, but the noise in these parameters is more adapted for gate bootstrapping and not for our case. It will not let us do a lot of additions. And since the other parameters provide a security that is less than the recommended parameters, we decide to select our specific parameters as defined in Table 6.2. We are aware that

Solutions	**N**	σ	λ
[19]	1024	2^{-25}	129
[86]	1024	2^{-30}	108
[6]	1024	10^{-9}	108
Our k-means with FHE	2048	10^{-15}	134

TABLE 6.2 – Security parameters and the security provided

these parameters will generate greater execution times.

6.5.2.1 Parameters choice procedure

As we have seen in the previous section, the values of the centers are integers where the parameter τ is a precision parameter and v is a scaling parameter.

Theoretically, τ must be enough big to provide the precision needed, but in practice we should also take care about the machine's limitations in order to avoid losing precision by dividing the values by τ. In the other hand, the parameter v allows to have intermediate results in the torus domain. Hence, we can choose to have a $v \geq ||x_i||$ parameter.

These assumptions have been confirmed empirically (see Figure 2) for Iris Dataset. As we can notice, with $\tau < 1000$ we do not have encouraging results. We see also that for $v = 4$ and $\tau = 1000$, we have an adjusted rand score of 1 which means that we have exactly the same results as the clear version of k-means. However, these graphs are specific to Iris dataset.

Our goal is not to have an ARI of 1 but an ARI that is closest to 1 as much as possible.

6.5.3 Performance results

The performance has been evaluated according to three metrics : efficiency, execution time and security. The proposed solution should have results as close as possible to the results provided by k-means in clear settings. In addition, it is important to have runtimes that are practical in real life.

6.5.3.1 Efficiency

To evaluate efficiency, we use two types of evaluation. First, we start with an external evaluation using the adjusted rand index and normalized mutual information to compare the result of our

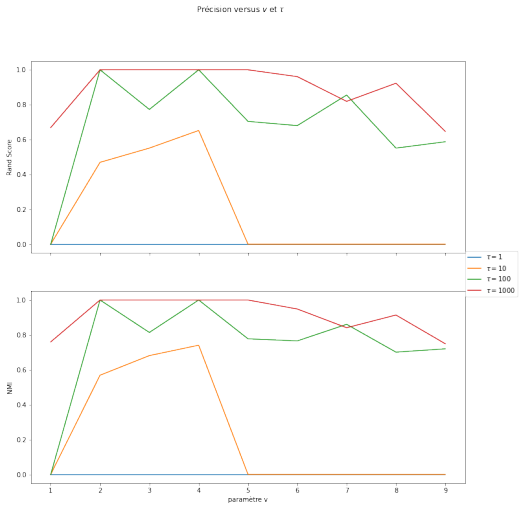


FIGURE 6.2 – Precision according to τ et v

algorithm to the standard k-means implemented in scikit-learn. Then, we proceed with an internal evaluation using a silhouette coefficient and the inertia.

Table 6.5 compares the results of the sklearn version of k-means and the HE version according to the adjusted rand score and the normalized mutual information.

Datasets	**Nb Iter (sklearn)**	**ARI**	**NMI**
Iris	5 (5)	1.0	1.0
Wine	9 (10)	0.98	0.97
Glass	10 (10)	0.97	0.96
Ovarian	4 (4)	1.0	1.0
Breast Cancer	9(9)	1.0	1.0
MiceProtein	17 (15)	0.92	0.93
PenDigits	30	0.93	0.92

TABLE 6.3 – HE-k-means version precision ($\tau = 10000$)

As depicted here, the results in HE are close to the standard k-means. In some datasets, we reproduce the exact clusters. As we have seen in the previous section, the results depend on the choice of parameters τ and v. In our experiments, we illustrated here only the better parameters by following the best practices described in the design section. We notice also that the number of iterations required to get the results is the same as the standard k-means.

However, the results that are not totally equal to the standard k-means ones are generally caused

by the precision loss due to the parameters. Thus, the error due to TFHE causes some precision loss, but this error affects only the observations in the borders of the clusters.

Since sometimes the results are not totally equal to the results provided by scikit-learn, we proceed to an internal evaluation in order to see if the HE results are better than the ones provided by sklearn. As shown in Table 6.4, by using the inertia and the silhouette score, we can confirm that the results are close to the standard k-means ones, and sometimes the results are even better like with the Wine dataset.

Datasets	**HE-kmeans**		**sklearn**		**Labels**	
	Inertia	**Sil**	**Inertia**	**Sil**	**Inertia**	**Sil**
Iris	6.98	0.5	6.98	0.5	7.80	0.45
Wine	48.96	0.30	48.97	0.299	49.99	0.29
Glass	19.22	0.365	19.20	0.365	36.13	-0.05
Ovarian	367.04	0.439	367.04	0.439	459.23	0.333
Breast	215.83	0.385	215.83	0.385	228.23	0.339
MiceProtein	971.42	0.135	971.13	0.136	-	-
PenDigits	3594.47	0.28	3584.97	0.28	5107.66	0.18

TABLE 6.4 – Intern evaluation

6.5.3.2 Execution time

In Table 6.5, we provided the execution time for each dataset as it is measured during our experiments. As we can see, the execution time is affected by the number of clusters. This can be explained by the construction of the delta matrix which takes a lot of time when achieving a bootstrapping, but also by the manipulation of a matrix instead of a vector to do comparisons. Indeed, the complexity of one iteration in the standard k-means is $O(n \times d \times k)$ but in our case the complexity is $O(n \times d \times k^2)$.

We can see that, for a fixed number of clusters and for the same number of observations, the number of attributes doesn't impact significantly the execution time. We can confirm that by comparing the execution time between Iris and Wine datasets, or between Ovarian and Breast Cancer datasets. The same thing happens for the number of observations. Hence, the complexity of our solution is $O(n)$ according to the number of observations.

The execution time for a dataset with a few number of clusters is practical in the real life. But, for a big dataset with a big number of clusters (like PenDegits dataset), the execution time is not practical when the execution is sequential. In this case, the parallelization is required. In fact, by parallelizing

the design of the delta matrix, the execution time is improved (see Table 5) .

Datasets	Seq Exec	8 threads
Iris	80s	19s
Wine	175s	59s
Glass	2853s	250s
Ovarian	69s	19s
Breast Cancer	183s	55s
MiceProtein	15187s	6289s
PenDigits	531222s	82958s

TABLE 6.5 – HE-k-means Execution time

6.5.3.3 Security

We can evaluate the security of this algorithm formally. The scheme used in this solution is actually sure and by choosing the parameters carefully, we ensure a security that is greater than the actual recommendation ($\lambda > 128$).

We have only one information leakage about the centroids. But, our goal here is to assign the observations to clusters. So, the leakage can be non-significant, but it can be avoided using other techniques than HE. We can use differential privacy as an example, but this technique is out of the scope of our study.

We can say that this solution is sure in a semi honest model. We need to be certain that the executed algorithm is the k-means algorithm. This can be verified using other cryptography techniques like "verifiable computing".

6.5.3.3.1 Discussion We can address our solution according to many aspects. First, the practicability of our solution, as shown by our experimentation, the proposed solution can be used in a cloud setting, and it is practical for datasets with few clusters. We showed that by using CPU parallelism, we can accelerate our solution to be more practical. Since building the delta matrix takes a lot of time due to bootstrapping operations and as stated before, this design can be completely parallelized to reduce execution time using GPUs.

Then, the results provided by our solution are similar to the standard k-means, so it can replace the k-means method in situations where privacy is required.

Generally, it is relevant to compare our solution with existing ones. However, in the limit of our efforts, there is no implementation shared by its authors. So, we limit our solution to compare the solutions formally. From the interactivity point of view, our solution is the first k-means implementation that performs the whole assignment step in an encrypted way and without interaction. The only interaction that is required in our solution is to carry out the division. We have as many divisions as the number of iterations.

6.6 CONCLUSION

In this work, we proposed a homomorphically encrypted $k-$means solution. Our main contribution is related to the assignment step that we can do completely in a secure way without any interaction between the client or other third party. In fact, it is the first practical solution that performs all the assignments in HE and without intermediate decryption. As a perspective to this work, we aim to parallelize our solution using GPUs as well as using other HE techniques to secure the k-means.

Chapitre 7

Developing Adaptive Homomorphic Encryption by Exploring Differential Privacy Technique

7.1 Introduction

In response to the vulnerabilities inherent in cloud systems, this study delves into a series of pioneering cryptographic techniques known as Privacy-Preserving Technologies (PPTs). These technologies are designed to enhance utility by harnessing advanced technologies such as cloud computing and machine learning while maintaining stringent privacy standards. The methodology employed involves post-processing the output of the decryption function using a mechanism that aligns with an appropriate differential privacy (DP) concept, introducing a noise that is proportional to the worst-case error expansion of the homomorphic computation.

In the context of implementing privacy-preserving mechanisms, the evaluation of the distribution of introduced and corresponding noises is paramount, particularly following the conventional analysis of homomorphic encryption noise, due to the potential insignificance of the noise relative to the original message. Ensuring the stability of privacy, in conjunction with the HE-DP amalgamation and relevant protocol, is critical for "Approximate Fully Homomorphic Encryption" where the retention of noise as the least significant bits of the final output during decryption is intentional.

The primary goal of integrating HE-DP schemes is to fortify machine learning applications by adding an extra layer of security against unorthodox adversaries. This research establishes a connection between the noise in Homomorphic Encryption and Differential Privacy, investigating this relationship

when the noise in Homomorphic Encryption is perceived as a database-dependent output perturbation. The paper introduces groundbreaking insights into the guarantees of Differential Privacy through this database addition.

7.2 Research Motivation

Several recent research endeavors have played a crucial role in aligning the potentials and impacts of homomorphic encryption and differential privacy, serving as the motivation for this study. Xiangyun Tang et al. [87] were pioneers in establishing a well-defined interchangeable property for ML classifiers and ML models in alignment with HE and DP. They introduced Heda, an innovative amalgamation of HE and DP, designed as a flexible switch to manage the privacy budget and precision parameter tuning to balance the inherent trade-offs. Homomorphic encryption is a crucial cryptographic technique that enables computations to be performed directly on encrypted data, thus removing the need for decryption. It serves as a robust solution for preserving the privacy and confidentiality of sensitive information while allowing computations on such encrypted data [88]. This method ensures data security even during processing, reducing dependence on trusted third parties and minimizing the risk of exposing sensitive information to potential threats. As described in [89], a homomorphic map preserves structure, meaning an operation on plaintexts corresponds to an operation on ciphertexts. This implies that altering the sequence of operations preserves the outcome post-decryption, i.e., 'encrypt-then-compute' and 'compute-then-encrypt' yield equivalent results.

Aligned with the trend of parametric models, Bossuat et al. (2022) [78] optimized their model of approximate homomorphic encryption, reducing the occurrences of failures and enhancing precision through distributed and random encapsulation, initiated with bootstrapping. Preliminary investigations have uncovered additional hybrid cryptographic techniques, which integrates intriguing multi-party primitives with the HE protocol in lattice-based cryptography.

Motivated by these advancements, this work pursues the design and validation of a refined HE-DP model, incorporating sensitivity analysis to ensure balanced noise insertion to mitigate various trade-offs. The motivation is driven by the desire to explore and amplify the synergistic capabilities of homomorphic encryption and differential privacy contributing to the progressing field of cryptographic research.

7.3 Relevant Mathematical Perspectives

The proposed model envisages the combination of a given approximate FHE scheme with the tool of differential privacy. The construction can be given as follows :

Given an approximate FHE scheme, we modify the decryption function by post-processing its output (similar to decrypted messages) with an appropriate chosen Differential privacy model. Here, we can consider two levels :

Approximate FHE with a static noise : This instance describes that the bound can be truly computed as a function of homomorphic encryption. Finally, this could be used on the input ciphertext.

Approximate FHE with a dynamic noise : In this case, the bound can be computed by the decryption function. In turn, the decryption function allows the input, the ciphertext, and the secret key.

We deploy the proposed model HEDP where we analyze a DP model while adding Gaussian dynamic noise to the input. Hence, let $\mathcal{F} = (KeyGen, Enc, Dec, Eval)$ be an FHE scheme with a plaintext space : $S \subset \mathbb{Z}$, where $S \subseteq \mathbb{Z}$ is the normal space with the normal $\|.\|$ operator.

We also consider a policy to add Gaussian noise with this text space and normal $\|.\|$ operator. We also assume a standard deviation norm to add Gaussian noise at a later stage to test the level of privacy. Therefore, for any deviation $\delta > 0$, $n \in N$, the *Kullback-Leibler Differential Privacy [90]* could be a dynamic method to add the noise. However, for the presented model, we follow the standard Gaussian noise insertion mechanism.

7.3.1 Gaussian Noise : Maintaining Privacy and Preserving Statistical Properties

Gaussian noise can be used as a mechanism to add differential privacy to a dataset. Gaussian noise is a type of random noise that follows a normal distribution, and it can be added to numerical data in order to mask the original values while still preserving the statistical properties of the data. In the context of differential privacy, Gaussian noise can be added to the data in order to make it more difficult to identify individual records or extract sensitive information from the dataset. The amount of noise added can be controlled by adjusting the standard deviation of the Gaussian distribution with higher standard deviations resulting in more noise and greater privacy protection. However, adding

too much noise can make the data less useful for analysis or modelling. So, it is important to strike a balance between privacy and utility when using this technique. For the use case, a standard deviation of 0.1 is used for generating noises. This value can be increased further but will result in a significant decrease in terms of accuracy.

7.4 Paillier Cryptosystem : Scheme and Properties

The Paillier cryptosystem is a public key cryptosystem used for the encryption and decryption of data. It is an asymmetric system, meaning that it employs two keys : a public key and a private key. The public key is utilized for encryption, while the private key is used for decryption. The Paillier cryptosystem possesses several distinctive properties, including its capability to perform homomorphic addition and scalar multiplication. This implies that it can execute addition and multiplication operations on ciphertexts without the necessity to decrypt them initially.

The system is semantically secure, which means that an attacker cannot learn any information about the plaintext from the ciphertext. For the study, the Paillier Cryptosystem is leveraged by the HEDP algorithm to ensure secured data transfer and computations with privacy. In the following section, two separate high-level descriptions of HEDP (e.g., client & server architecture) are presented.

7.4.1 Client-Side Algorithms and Server-Side Algorithms

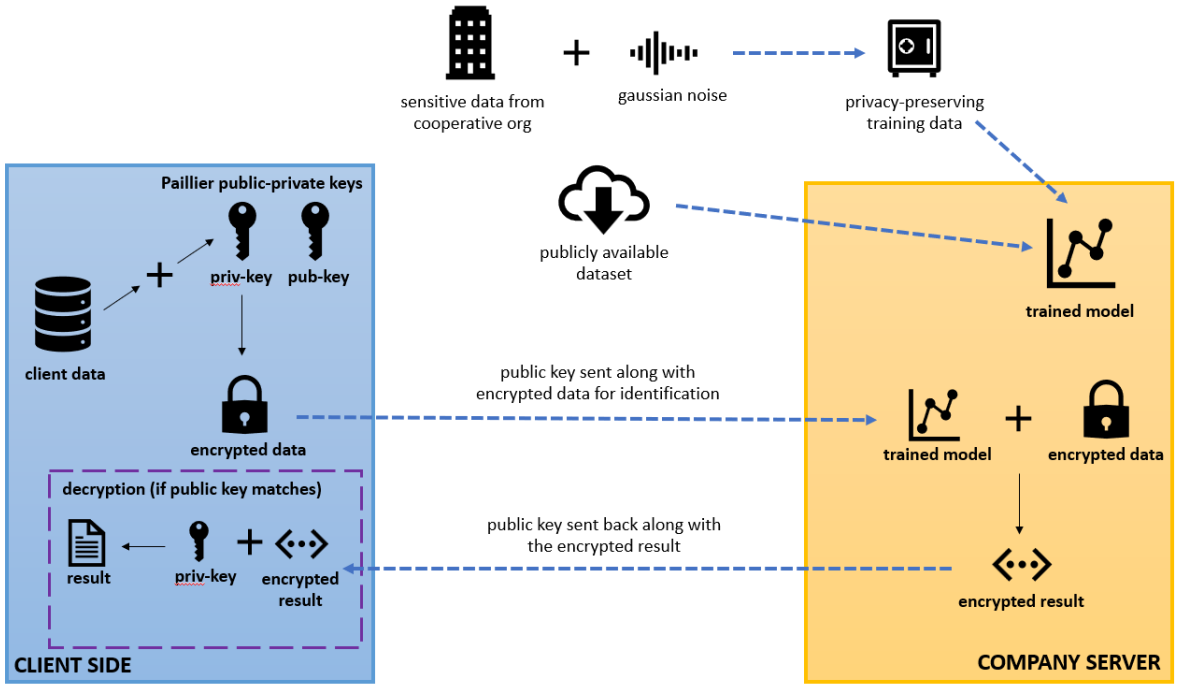


FIGURE 7.1 – Client-Server HEDP interaction Schema

Algorithm 5 Process :

generate Paillier public-private key pair using Paillier Cryptographic Scheme
save the public-private key
create a PaillierPublicKey object *pub_key* with the value of n from the public key
create a PaillierPrivateKey object *priv_key* with *pub_key*, p, and q from the private key.
for each data in dataset **do**
    add Gaussian noise at randomized intervals
    encrypt the data with *pub_key*
end for
convert the encrypted data
append *pub_key*
send the data → **SERVER** side
while no data prediction from server **do**
    wait
end while
if *pub_key* of response == generated *pub_key* **then**
    for data in prediction **do**
        decrypt the data with *priv_key*
    end for
end if

Algorithm 6 Process :

```
while no data request from client do
    wait
end while
Load trained weights from base model
for data in data request do
    if data == pub_key then
        Create a PaillierPublicKey object pub_key with the value of n from pub_key
    else
        Convert data → Paillier EncryptedNumber objects
        prediction ← dot product of loaded weights and EncryptedNumber objects
        result ← prediction , append pub_key (data)
    end if
end for
send the result data → CLIENT side
```

7.5 Experimental Validation and Discussion on Results

Combining homomorphic encryption and differential privacy provides an even more powerful tool for secure machine learning. By combining these two techniques, it is possible to perform computations on encrypted data while also ensuring that the privacy of the individuals in the data set is protected. In this work, a novel measurement is proposed and an evaluation of the performance of Homomorphic Encryption with Differential Privacy (HEDP) has been done on a Breast Cancer Wisconsin dataset.

7.5.1 Proposed HEDP : Architecture, Process and Code Walkthrough

Based on the dataset, the objective is to predict whether the type of cancer is Malignant or Benign based on the attributes provided in the dataset. It is to be kept in mind that the flow of machine learning model training and prediction should proceed in such a manner that it preserves both privacy and security. To ensure the maximum security and privacy, HEDP algorithm is proposed. The methodology starts by training a base model on a publicly available dataset. This dataset can be used by any organization or individual who wishes to train an ML model. However, if the dataset is sourced from a sensitive organization, then privacy-preserving measures need to be taken. This is where differential privacy comes in. Gaussian noise is added to the sensitive data to preserve privacy. The amount of noise added depends on a privacy parameter, which is set by the organization or individual providing the sensitive data. Once the privacy-preserving modifications are made to the dataset, the

base model is trained on this modified dataset. This base model is used by any client who wants to make predictions on their own sensitive data. However, the client does not wish to reveal their sensitive data to the party (company offering ML services). To further enhance the privacy, Homomorphic Encryption is added into the mix, particularly the Paillier encryption scheme. The client generates a Paillier public-private key pair and uses the public key to encrypt their data. The public key is then attached along with the data and sent to the ML company over a secure network. Once the ML component receives the encrypted data, it uses the public key to extract Paillier encrypted objects. Then it performs machine learning computations on the Paillier objects using the already trained base model to predict the output. The output is an encrypted prediction. This is because the encrypted data is encrypted with the Paillier cryptosystem, which is a homomorphic encryption scheme. Any sort of addition and scalar multiplication done on this encrypted data is the same as doing the same operations on the decrypted data. The encrypted prediction is sent back to the client. The client then decrypts it with their Paillier private key and gets their result.

7.6 Scopes of Implementations

In this experiment, the dataset used is Breast Cancer Wisconsin. The dataset is publicly available, for the demonstration purposes, it is assumed that the data received is pre-processed by injecting Gaussian noise. To increase the strength of the Gaussian noise added to the dataset, it is required to increase the value of the standard deviation (σ) parameter of the Gaussian distribution used to generate the noise. This parameter controls the spread of the distribution, so higher values of σ will generate noise with greater magnitude. Different values of σ can be experimented to find the optimal level of noise for the specific use case. However, it is crucial that adding too much noise may adversely affect the accuracy of the given machine learning model. The following sub-section describes the deployment process for both client and server architecture, which is essential for experimental validation.

7.6.1 Client-Side Code Walkthrough

The following lists of functions describe the client-side deployment :

— **storeKeys()** : This function generates public and private keys based on the Paillier Homomorphic Encryption Scheme, and saves the keys in a data file named *client_public_private_keys.data*.

— **getKeys()** : This function reads the *client_public_private_keys.data* file and creates public and private key objects using the Paillier module.

— **serializeData()** : This function takes the *public_key* object and data as inputs, encrypts the data using the public key, and returns a serialized data object. The encrypted data is stored as a list of tuples with the ciphertext and exponent. The encrypted data sent to the server appears as follows :

— **load_prediction()** : This function reads a data file named *prediction.data* and returns its contents as a dictionary.

The contents of the *client_public_private_keys.data* and *prediction.data* files are illustrated below.

The **load_prediction()** function then loads a pre-generated encrypted prediction from the server and decrypts it with the private key of the Paillier scheme. The resulting answer is a Regressed Result. This regressed result is then passed through a Sigmoid function that strictly compresses the regressed result between 0 and 1.

7.6.2 Server-Side Code Walkthrough

Firstly, a base model is trained on the privacy-preserved data (in our case). There will not be much difference in training the base model. The process is the same. Once the model is trained, the trained weights are saved and used for homomorphic computations.

The *trained_weights* contain the weights fitted by the model on the breast cancer dataset. The following lists of responsible functions are mandatory :

— **getData()** : retrieves the encrypted data from the client and loads it into the system.

— **computeData()** : This function uses the *trained_weights* to perform a dot product with encrypted Paillier objects. This is a homomorphic scalar multiplication, a valid operation on the Paillier encryption scheme. However, the result is not in binary format and requires conversion to binary using the sigmoid function. It is important to note that the Paillier Encryption Scheme is a Partially Homomorphic Scheme because it cannot handle division homomorphism. Even mimicking the division with multiplicative inverse results in inaccurate encrypted notation. Therefore, to obtain accurate results, the sigmoid function is employed on the client side.

— **computeAndSerializeData()** : formats the encrypted results along with the respective public

key.

— **save_prediction()** : saves the prediction in the *prediction.data* file. This is the predicted result sent to the client for decryption.

7.6.3 Performance Analysis : Proposed HEDP versus Standard Algorithms

The analysis comprises of the plot for Accuracy vs Iterations.

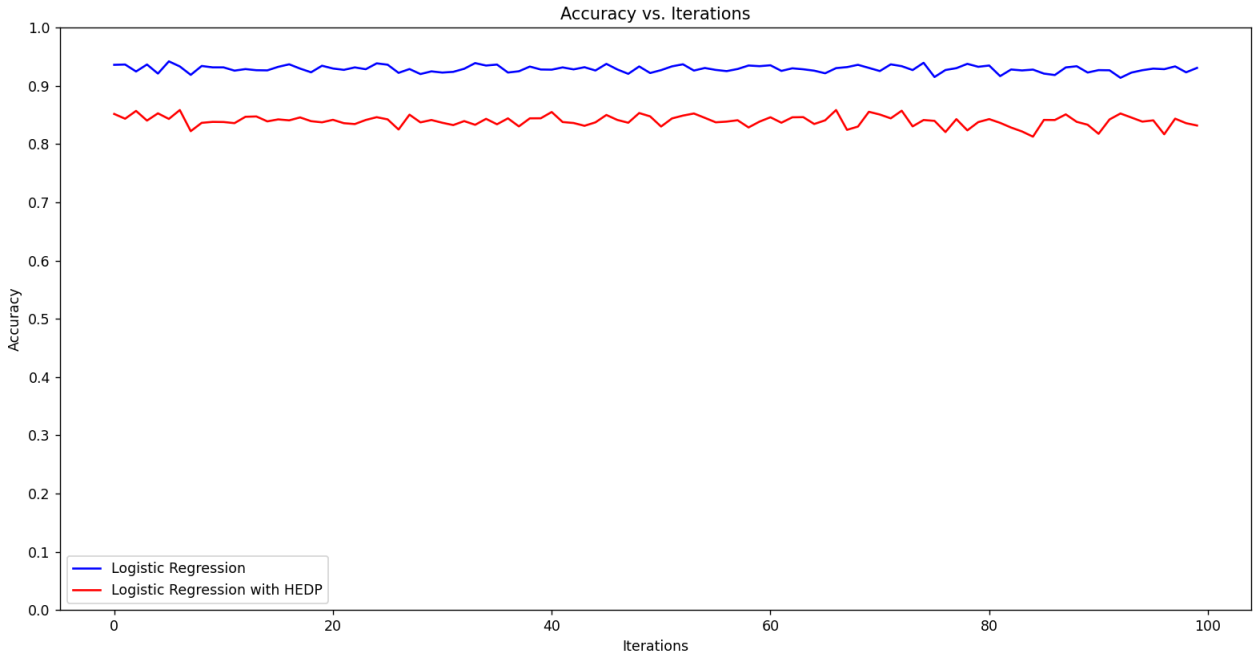


FIGURE 7.2 – Accuracy Versus Iteration Plot

Figure 7.2 demonstrates the accuracy versus iteration plot. It is demonstrated that the observed loss is not significant since the training and prediction methods did not use any lossy encryption schemes. The accuracy decline in HEDP can be attributed to the noisy dataset used for training. However, it is noteworthy that the model performed remarkably well, considering the fact that it was trained entirely using privacy-preserving techniques. During the execution phase, the CPU consumptions from both the sides (client and server) were observed.

7.6.3.1 The Client-side plot

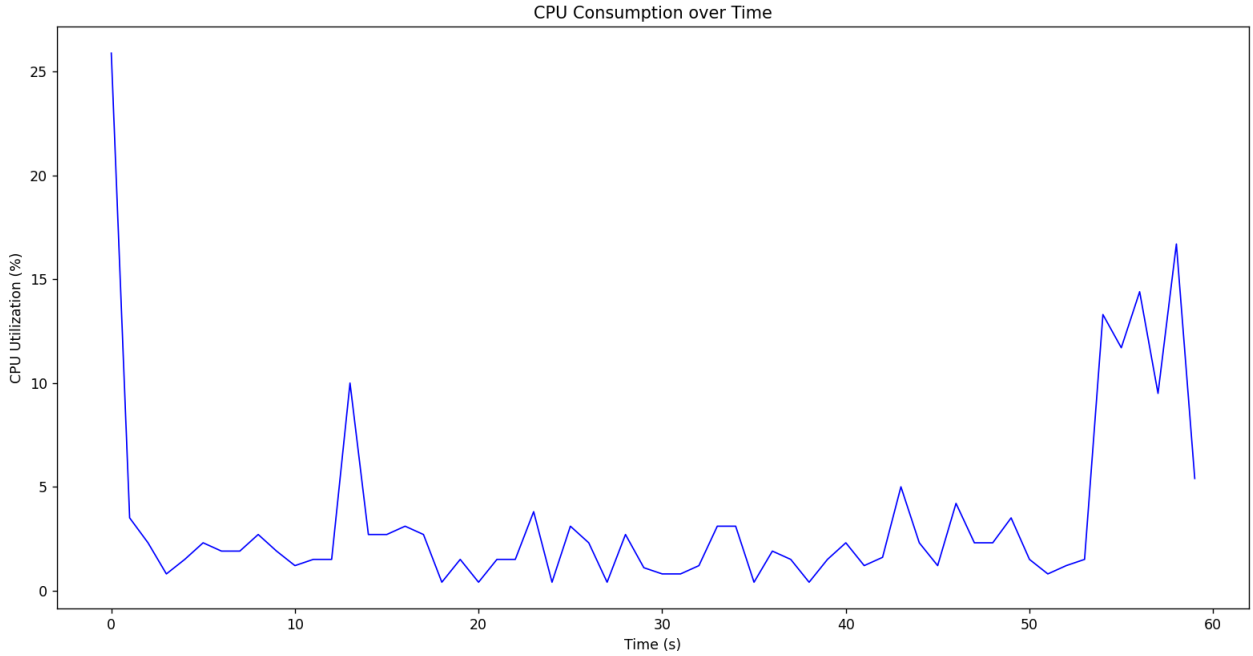


FIGURE 7.3 – CPU Utilization with Time occupancy (Client Side)

Figure 7.3 shows the CPU utilization with time occupancy on the client side.

Observations : The initial spike was due to Paillier encryption, where the client generated Paillier public, private keys and then used it to encrypt the data it intended to send to the server. The sudden spike at the end is attributed to decryption (using private key to decrypt the result) as well as the sigmoid operation on the received result. Here, the maximum CPU utilization observed is 25%.

7.6.3.2 Server-side Plot

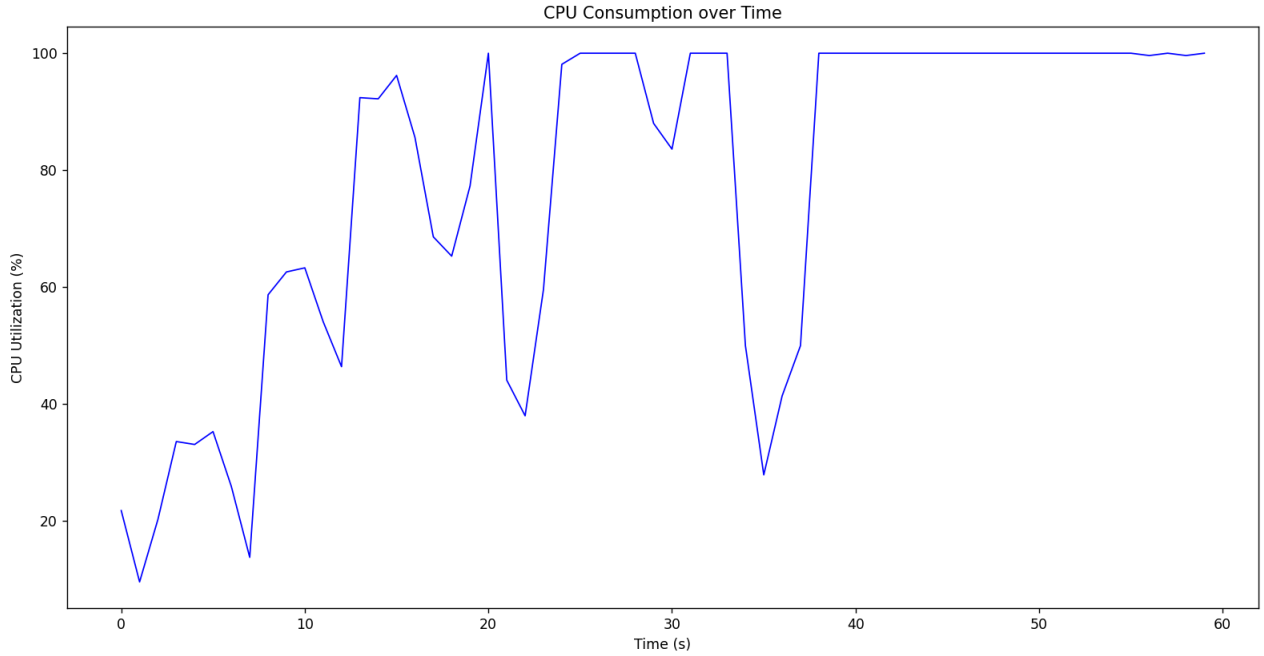


FIGURE 7.4 – CPU Utilization with Time occupancy (Server Side)

Figure 7.4 shows the CPU utilization with time occupancy on the server side.

Observations :

The server is idle initially. As soon as it received the data, the computational process kicked in (graph peaks consistently after the 35s mark). The computational process is heavy as it is evident in the plot. It rapidly consumed around 100% of CPU utilization for homomorphic operations on encrypted data. Thus, the maximum CPU Utilization reaches 100%.

7.6.4 Standard Algorithm (Linear Regression without HEDP) CPU Plot

The following plot, Figure 7.5, demonstrates the CPU utilization of the standard algorithm [15], which was trained on the same server.

Observations :

The training process lasted only a few seconds (7s). The peak CPU utilization is around 40%. This is much lesser than the HEDP counterpart. The CPU utilization drops quickly after the training process is complete, as there is no data transmission over the network involved.

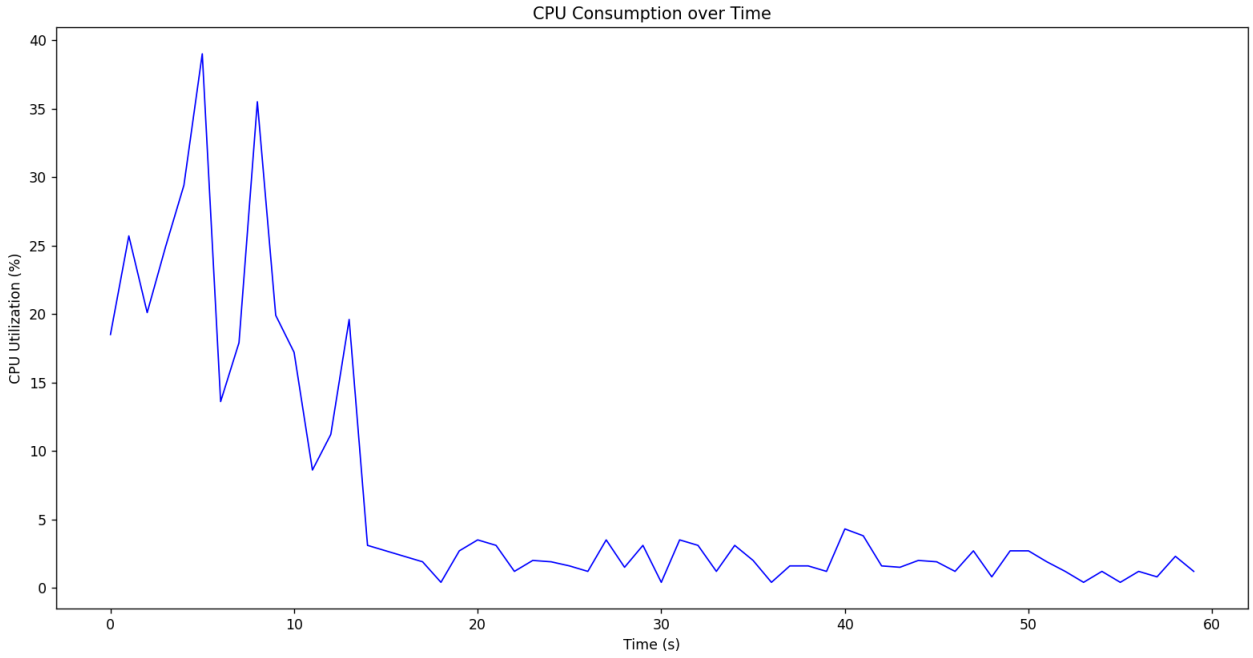


FIGURE 7.5 – Standard CPU Utilization without HEDP scenario

7.7 Sensitivity Analysis

Sensitivity analysis [91] is expressed as the relation to the Laplace mechanism privacy budget and the justified sensitivity to regulate the amount of noise addition. Therefore, every HEDP algorithm should deduce suitable mathematical expression to fix privacy budget formula and accommodate most likelihood sensitivity procedure for DP mechanism. Without the loss of generality, we assume that to guarantee the efficacy of Differential Privacy if considered homomorphically, then it is expected to

consider only the privacy measure of terminal output.

In few cases, the entropy of terminal output follows :

$$\alpha + N(0, \sigma^2) \quad .....(1)$$

where α is the "true" output of the algorithm. Thus, an initial approach may attempt to ensure ρ is large enough to mask the difference $(\alpha - \alpha')$ over adjacent databases.

In this work, we consider failure probability δ of DP under the real variance of HE model, where sensitivity δf could vary with the value of variance in proportion with a monotonically increasing function. However, the variance of an algorithm's output could be more prominent numerically when it is evaluated homomorphically (this behavior is dependent on the input data). Therefore, the proposed algorithm presented here is to model the output from the database with proper noise distribution. Thus, this sensitivity analysis comprises of database and noise analysis.

The following core theorem can be described as :

Let ε (privacy budget for the proposed HE-DP algorithm including client and server side deployment) $\in (0, 1)$ be arbitrary. For $c^2 > 2\ln(1.25/\delta)$, the Gaussian Mechanism is (ε, δ)-differentially private whenever $\sigma \geq c\delta f/\varepsilon$, where δf is the sensitivity.

We consider two databases PD (Pilot database) and RD (Reference database) and let A be the algorithm output when we use PD as input database, and A' when we use Reference database RD'.

Our strategy will be to show the ratio of probability density functions as :

$$\frac{f_{PD}(A(\alpha))}{f_{RD'}(A'(\alpha))} > e^{\varepsilon}, ...(2)$$

except with probability at most δ as α follows the distribution of the proposed HEDP algorithm A.

7.7.1 Exceptions in the proposed HEDP model

In the proposed model with HE, the multiplication will automatically be part of plain texts and thus it will impress on the variance of $(\alpha - \alpha')$. Therefore, we assume polynomial model as $P_{\sigma_1^2\sigma_2^2}$ and thus added entity with plaintexts t_1 and t_2 can be expressed as :

$$P_{\sigma_1^2\sigma_2^2} + \sigma_1^2\|t_1\|^2 + \sigma_2^2\|t_2\|^2, ...(3)$$

Therefore, for the given homomorphic encryption, input database should be influential with respect to the given entropy followed :

$$[\alpha + N(0, \sigma^2)].....(4)$$

As we consider $(\alpha - \alpha')$ over adjacent databases, therefore to tune with the consulting databases focused for training, the objective is to fix the Gaussian mechanism for the proposed HEDP algorithm in such a way that the variance σ will depend on the combined value of (ε, δ) and it will be evaluated as the following :

$$2\ln\left(\sqrt{\text{prob}_1/\text{prob}_2}\right) > 1.....(5)$$

the value of ε also varies with noise scale. If we consider κ as coefficient which will support to express the agreement of the binary outcome of the two training databases, where the noise ε is added to achieve differential privacy for different mathematical operations relevant to homomorphic encryption.

However, to find out the ratio of the probability of given noise function, we follow the following formula :

$$\frac{f_{PD}}{f_{RD'}} = \exp\left(\frac{\|\gamma - \alpha'\|_2^2}{2\sigma^2} - \frac{\|\gamma - \alpha\|_2^2}{2\sigma^2}\right).....(6)$$

$$= \exp\left(\frac{1}{2\sigma^2} + (\|\gamma - \alpha + \kappa\|_2^2 - \|\gamma - \alpha\|_2^2)\right).....(7)$$

$$= \exp\left(\frac{1}{2\sigma^2} + 2(\gamma - \alpha) \cdot \kappa + \|\kappa\|_2^2)\right).....(8)$$

It signifies that the database for training with deliberate noise needs a Pilot database (PD), where the proposed algorithm A and its variance A' will be functional to another parallel reference database with a relational expression as : $A \sim N(\alpha, \Sigma)$, here Σ can find its contemporary diagonal element Σ'.

Thus,

$$\frac{f_{PD}}{f_{RD'}} = \prod_{i=1}^{PD} \frac{\sigma_{RD}}{\sigma_{PD}} \exp\left(\frac{1}{2}\left(\frac{\gamma_i - \alpha_i'}{\sigma_{RD}}\right)^2 - \left(\frac{\gamma_i - \alpha_i}{\sigma_{PD}}\right)^2\right).....(9)$$

Hence, with log sensitivity and iterations for the variance, we will be interested to evaluate maximum likelihood probability and we can rewrite the expression as :

$$\frac{f_{PD}}{f_{RD'}} = \prod_{i=1}^{PD} 1/\zeta_i \exp\left(1/2 \sum_{i=1}^{PD} \zeta_i^2 \left((\gamma_i - \alpha_i)/\sigma_i - \kappa_i\right)^2 - ((\gamma_i - \alpha_i)/\sigma_i)^2.....(10)\right.$$

This is in tune with polynomial model of $P_{\sigma_1^2 \sigma_2^2}$ for HE multiplication. It also highlights that in the present simplistic model rescale and squaring have not been addressed. After inclusion of these

components of noise growth, the maximum likelihood probability of log variance could be changed accordingly.

Here, ζ_i is the relation between diagonal entries for the pilot and referential database, subjected for the training and sensitivity κ should be in the appropriate range of variance of $(\alpha - \alpha')$.

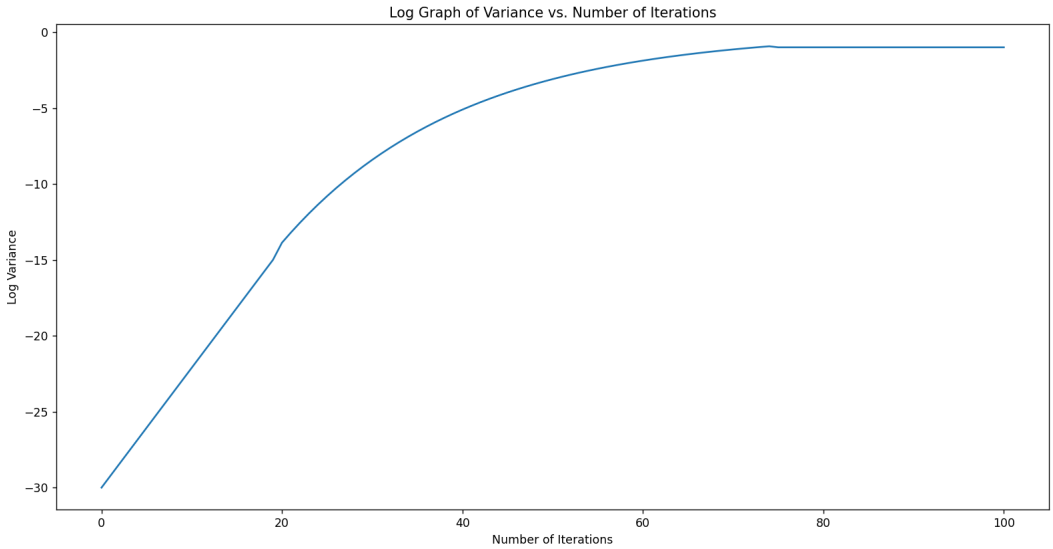


FIGURE 7.6 – Sensitivity Analysis of Proposed HEDP Algorithm - Log Variance with Iterations

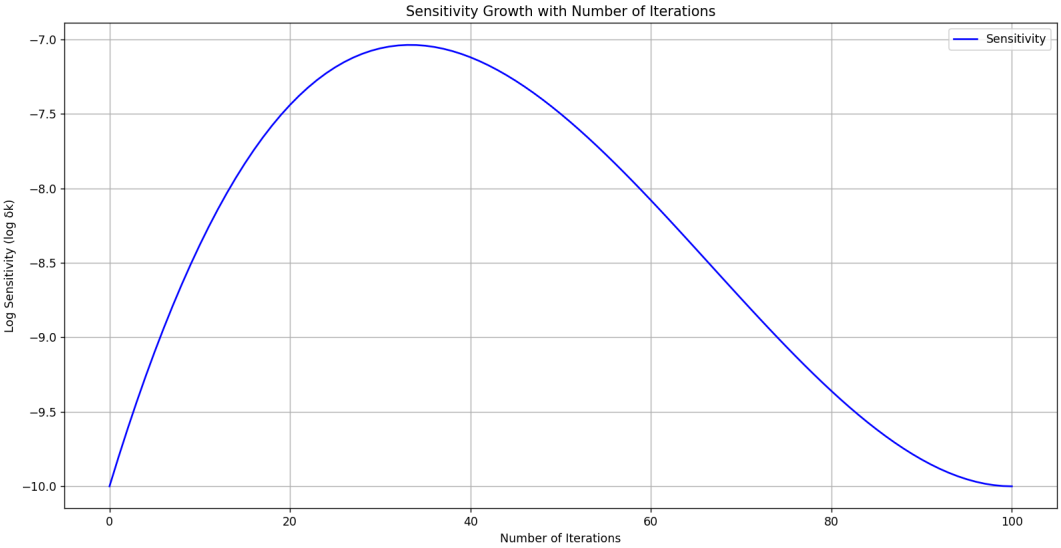


FIGURE 7.7 – Sensitivity Analysis of Proposed HEDP Algorithm - Log Sensitivity with Iterations

Following the formulated eq. (8), (9) (10) above, the validation of sensitivity analysis for proposed

HEDP algorithm can be done. We observe that there are two phases to measure the trend of the system : a). log variance with iterations. b). log sensitivity with iterations.

Significantly, in the first plot Figure 7.6, it is observed that log variance inclines towards positive training value after certain number of iterations. Thus, training accuracy with noise becomes more consistent with positive value as it grows up with number of iterations. However, proposed HEDP in Figure 7.7) demonstrates to attain a peak value with log sensitivity scale -7 in 40th iterations. The curve flattens more as it grows up with a greater number of iterations. It signifies that trade-off between accuracy and computational speed up in the proposed model becomes stable with the present use-case. However, to find out more robust sensitivity, the value of Gaussian noise, the noise variance and dependency analysis have also to be performed on the HEDP algorithm.

Conclusion and Scope of Further Research

In this work, a rudimentary algorithm for combining homomorphic encryption and differential privacy is proposed to demonstrate an effective strategy towards precision, computational efficiency and privacy budget trade-off. It is emphasized with the deployment of such a proof-of-concept demonstrated with an elementary client-server architecture, which could position suitable justification of library compatibility for HE environment. We observe that, in DP, the main challenge is to reduce the tradeoff between privacy and accuracy. To address this challenge, future research can explore the development of more robust mechanisms that add less noise while providing more privacy guarantees. This is indicative for complete sensitivity analysis of such HEDP protocol to perform noise analysis and to check message dependency.

Additionally, it is possible to propose different relaxations for DP in the context of distributed paradigm or to boost

General Conclusion

The present thesis has undertaken a comprehensive exploration of the implications of employing homomorphic encryption in the domain of Machine Learning as a Service (MLaaS) and the protection of sensitive data. The rapid expansion of MLaaS has raised significant concerns about the confidentiality of sensitive data, thus prompting our investigation into homomorphic encryption as a viable solution. Extensive research has been conducted on its application across various aspects of machine learning, including its deployment in a multi-cloud environment, its integration into the k-nearest neighbors (k-NN) algorithm, and its adaptation to the k-means clustering algorithm.

The findings of this thesis have conclusively demonstrated that the use of homomorphic encryption can effectively ensure the security of sensitive data while enabling complex data processing operations. Our study has revealed that despite the complexities associated with implementing homomorphic encryption, its performance remains comparable to that of unencrypted methodologies, thereby presenting compelling opportunities for strengthening the protection of sensitive data in the realm of cloud computing.

These promising discoveries open up stimulating avenues for future scholarly investigation. Enhancing the hardware components of homomorphic encryption schemes is a prospect worth considering to bolster processing capabilities. Additionally, a thorough examination of the integration of homomorphic encryption with complementary privacy methodologies, such as differential privacy, could further enhance the security of machine learning models in sensitive paradigms.

In summary, this thesis represents a substantial contribution to advancing knowledge about securing sensitive data within MLaaS environments. The findings serve to stimulate ongoing exploration and broader implementation of homomorphic encryption, thereby unlocking new possibilities for securing sensitive data within machine learning and cloud computing applications.

Many challenges need to be addressed to apply privacy-preserving machine learning in real-world applications. Although the standards, platforms, and implementations of homomorphic encryption described in this chapter contribute to the advancement of Homomorphic Encryption in Machine Learning (HEML), there are still specific challenges to be tackled, including overhead, performance, interoperability, bootstrapping bottlenecks, sign determination, and common frameworks :

— **Overhead** : Compared to its unencrypted counterpart, HEML comes with significant overhead, making it unsuitable for many applications. However, for non-HE models, the training phase of ML involves a computationally intensive effort. Even with modern techniques, it becomes increasingly challenging with HE. A recent trend is to bypass the training step by using pre-trained models to strike a balance between complexity and accuracy.

— **Parallelization** : Incorporating well-established and novel algorithms is one approach to deal with the computational overhead. High-performance computers, distributed systems, and specialized resources can all be utilized in HEML models. Multi-core processing units (GPUs, FPGAs, etc.) and customized chips (ASICs) provide more efficient and user-friendly HEML environments. Another approach to improve overall efficiency is batching and parallelizing numerous bootstrapping operations.

— **Comparison and min/max function** : New methods are needed to compare numbers encrypted by Homomorphic Encryption (HE). Currently, comparison and min/max functions are evaluated using Boolean functions where input numbers are encrypted bit by bit. However, bit-wise encryption methods require relatively expensive computations for basic arithmetic operations such as addition and multiplication.

— **PPML (Privacy Preserving Machine Learning) tools** : Designing a high-performing and secure PPML solution without a thorough understanding of HE is practically challenging for the deployment of these technologies. PPML developers need expertise in both machine learning and security. PPML, which utilizes HE, has not been widely accepted by the ML community due to the high entry barrier of HE and the lack of user-friendly tools.

— **Hybrid protocols** : Adopting hybrid protocols, which combine two or more protocols to leverage their advantages and avoid their disadvantages, is a promising direction for performance improvements.

Publications

1. [*Published*] Y. Ameur, S. Bouzefrane, and V. Audigier. "Application of Homomorphic Encryption in Machine Learning". In : Daimi, K., Alsadoon, A., Peoples, C., El Madhoun, N. (eds) Emerging Trends in Cybersecurity Applications. Springer, Cham. https://doi.org/10.1007/978-3-031-09640-2_18.
2. [*Published*] Y. Ameur, R. Aziz, V. Audigier, and S. Bouzefrane. "Secure and Non-interactive k-NN Classifier Using Symmetric Fully Homomorphic Encryption". In : Domingo-Ferrer, J., Laurent, M. (eds) Privacy in Statistical Databases. PSD 2022. Lecture Notes in Computer Science, vol 13463. Springer, Cham. https://doi.org/10.1007/978-3-031-13945-1_11.
3. [*Published*] Y. Ameur, S. Bouzefrane, and L. V. Thinh. "Handling security issues by using homomorphic encryption in multi-cloud environment". Procedia Computer Science, Volume 220, Pages 390-397, 2023. https://doi.org/10.1016/j.procs.2023.03.050.
4. [*Published*] Y. Ameur. "Developing Adaptive Homomorphic Encryption by Exploring Differential Privacy Technique". Journal of Cyber Security and Mobility, 2023.
5. [*InProgress*] Y. Ameur. "Secure k-means clustering using TFHE".

Bibliographie

[1] Statista. *Number of Internet of Things (IoT) connected devices worldwide from 2015 to 2025*. https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/. Accessed : April 8, 2023. 2021.

[2] Yulliwas Ameur, Samia Bouzefrane et Thinh Le Vinh. "Handling security issues by using homomorphic encryption in multi-cloud environment". In : *The 14th International Conference on Ambient Systems, Networks and Technologies (ANT)*. Leuven, Belgium, mars 2023. URL : https://hal.science/hal-03933238.

[3] Yulliwas Ameur et al. "Secure and Non-interactive k-NN Classifier Using Symmetric Fully Homomorphic Encryption". In : *Privacy in Statistical Databases*. Sous la dir. de Josep Domingo-Ferrer et Maryline Laurent. Cham : Springer International Publishing, 2022, p. 142-154. ISBN : 978-3-031-13945-1.

[4] Yulliwas Ameur, Samia Bouzefrane et Soumya Banerjee. "Developing Adaptive Homomorphic Encryption through Exploration of Differential Privacy". In : *Journal of Cyber Security and Mobility* -.- (2023). En cours d'édition, p. -. ISSN : 2245-4578. DOI : -. URL : -.

[5] Gizem S Çetin et al. "Depth optimized efficient homomorphic sorting". In : *International Conference on Cryptology and Information Security in Latin America*. Springer. 2015, p. 61-80.

[6] Martin Zuber et R. Sirdey. "Efficient homomorphic evaluation of k-NN classifiers". In : *Proceedings on Privacy Enhancing Technologies* 2021 (2021), p. 111 -129.

[7] Yulliwas Ameur, Samia Bouzefrane et Le Vinh Thinh. "Handling security issues by using homomorphic encryption in multi-cloud environment". In : *Procedia Computer Science* 220 (2023). The 14th International Conference on Ambient Systems, Networks and Technologies Networks (ANT 2022) and The 6th International Conference on Emerging Data and Industry 4.0 (EDI40), p. 390-397. ISSN : 1877-0509. DOI : https://doi.org/10.1016/j.procs.2023.03.050. URL : https://www.sciencedirect.com/science/article/pii/S1877050923005859.

[8] Ilaria Chillotti et al. "TFHE : Fast Fully Homomorphic Encryption Over the Torus". In : *J. Cryptol.* 33.1 (2020), 34–91. ISSN : 0933-2790. DOI : 10.1007/s00145-019-09319-x. URL : https://doi.org/10.1007/s00145-019-09319-x.

[9] Pascal Paillier. "Public-key cryptosystems based on composite degree residuosity classes". In : *International conference on the theory and applications of cryptographic techniques*. Springer. 1999, p. 223-238.

[10] R L Rivest, L Adleman et M L Dertouzos. "On Data Banks and Privacy Homomorphisms". In : *Foundations of Secure Computation, Academia Press* (1978), p. 169-179.

[11] Craig GENTRY. "Fully homomorphic encryption using ideal lattices". In : *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*. Sous la dir. de Michael MITZENMACHER. ACM, 2009, p. 169-178. DOI : 10.1145/1536414.1536440. URL : https://doi.org/10.1145/1536414.1536440.

[12] *PALISADE Lattice Cryptography Library (release 1.11.5)*. https://palisade-crypto.org/. 2021.

[13] Abbas ACAR et al. "A Survey on Homomorphic Encryption Schemes : Theory and Implementation". In : *ACM Comput. Surv.* 51.4 (2018). ISSN : 0360-0300. DOI : 10.1145/3214303. URL : https://doi.org/10.1145/3214303.

[14] Marten van DIJK et al. "Fully Homomorphic Encryption over the Integers". In : *Advances in Cryptology – EUROCRYPT 2010*. Sous la dir. d'Henri GILBERT. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010, p. 24-43. ISBN : 978-3-642-13190-5.

[15] Craig GENTRY. "Computing Arbitrary Functions of Encrypted Data". In : *Commun. ACM* 53.3 (2010), 97–105. ISSN : 0001-0782. DOI : 10.1145/1666420.1666444. URL : https://doi.org/10.1145/1666420.1666444.

[16] Oded REGEV. "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography". In : *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*. STOC '05. Baltimore, MD, USA : Association for Computing Machinery, 2005, 84–93. ISBN : 1581139608. DOI : 10.1145/1060590.1060603. URL : https://doi.org/10.1145/1060590.1060603.

[17] Vadim LYUBASHEVSKY, Chris PEIKERT et Oded REGEV. "On Ideal Lattices and Learning with Errors over Rings". In : *J. ACM* 60.6 (2013). ISSN : 0004-5411. DOI : 10.1145/2535925. URL : https://doi.org/10.1145/2535925.

[18] Kurt R. ROHLOFF et David COUSINS. "A Scalable Implementation of Fully Homomorphic Encryption Built on NTRU". In : *Financial Cryptography Workshops*. 2014.

[19] Ilaria CHILLOTTI et al. *TFHE : Fast Fully Homomorphic Encryption Library*. https ://tfhe.github.io/tfhe/. August 2016.

[20] Oded REGEV. "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography". In : t. 56. Jan. 2005, p. 84-93. DOI : 10.1145/1568318.1568324.

[21] Léo DUCAS et Daniele MICCIANCIO. "FHEW : bootstrapping homomorphic encryption in less than a second". In : *Advances in Cryptology–EUROCRYPT 2015 : 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I 34*. Springer. 2015, p. 617-640.

[22] Ilaria CHILLOTTI et al. "TFHE : fast fully homomorphic encryption over the torus". In : *Journal of Cryptology* 33.1 (2020), p. 34-91.

[23] Mark A. WILL et Ryan K.L. KO. "Chapter 5 - A guide to homomorphic encryption". In : *The Cloud Security Ecosystem*. Sous la dir. de Ryan KO et Kim-Kwang Raymond CHOO. Boston : Syngress, 2015, p. 101-127. ISBN : 978-0-12-801595-7. DOI : https://doi.org/10.1016/B978-0-12-801595-7.00005-7. URL : https://www.sciencedirect.com/science/article/pii/B9780128015957000057.

[24] Yulliwas Ameur, Samia Bouzefrane et Vincent Audigier. “Application of Homomorphic Encryption in Machine Learning”. In : *Emerging Trends in Cybersecurity Applications*. Sous la dir. de Kevin Daimi et al. Cham : Springer International Publishing, 2023, p. 391-410. isbn : 978-3-031-09640-2. doi : 10.1007/978-3-031-09640-2_18. url : https://doi.org/10.1007/978-3-031-09640-2_18.

[25] Zhigang Chen et al. “Bibliometrics of Machine Learning Research Using Homomorphic Encryption”. In : *Mathematics* 9 (nov. 2021), p. 2792. doi : 10.3390/math9212792.

[26] Trevor Hastie, Robert Tibshirani et Jerome Friedman. “Unsupervised learning”. In : *The elements of statistical learning*. Springer, 2009, p. 485-585.

[27] Ralf Bender et Ulrich Grouven. “Ordinal logistic regression in medical research”. In : *Journal of the Royal College of physicians of London* 31.5 (1997), p. 546.

[28] Vernon Gayle, P Lambert et RB Davies. “Logistic regression models in sociological research”. In : *University of Stirling, Technical Paper* 1 (2009).

[29] XiaoFeng Wang et al. *iDASH secure genome analysis competition 2017*. 2018.

[30] Shuang Wu et al. “Privacy-preservation for stochastic gradient descent application to secure logistic regression”. In : *The 27th Annual Conference of the Japanese Society for Artificial Intelligence*. T. 27. 2013, p. 1-4.

[31] Yoshinori Aono et al. “Scalable and secure logistic regression via homomorphic encryption”. In : *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*. 2016, p. 142-144.

[32] Miran Kim et al. “Secure logistic regression based on homomorphic encryption : Design and evaluation”. In : *JMIR medical informatics* 6.2 (2018), e8805.

[33] Joppe W Bos, Kristin Lauter et Michael Naehrig. “Private predictive analysis on encrypted medical data”. In : *Journal of biomedical informatics* 50 (2014), p. 234-243.

[34] Payman Mohassel et Yupeng Zhang. “Secureml : A system for scalable privacy-preserving machine learning”. In : *2017 IEEE symposium on security and privacy (SP)*. IEEE. 2017, p. 19-38.

[35] Raphael Bost et al. “Machine Learning Classification over Encrypted Data”. In : *IACR Cryptol. ePrint Arch.* 2014 (2015), p. 331.

[36] Shafi Goldwasser et Silvio Micali. “Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information”. In : *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*. STOC ’82. San Francisco, California, USA : Association for Computing Machinery, 1982, 365–377. isbn : 0897910702. doi : 10.1145/800070.802212. url : https://doi.org/10.1145/800070.802212.

[37] Frank Li, Richard Shin et Vern Paxson. “Exploring Privacy Preservation in Outsourced K-Nearest Neighbors with Multiple Data Owners”. In : *Proceedings of the 2015 ACM Workshop on Cloud Computing Security Workshop*. CCSW ’15. Denver, Colorado, USA : Association for Computing Machinery, 2015, 53–64. isbn : 9781450338257. doi : 10.1145/2808425.2808430. url : https://doi.org/10.1145/2808425.2808430.

[38] Bharath K. Samanthula, Yousef Elmehdwi et Wei Jiang. “k-Nearest Neighbor Classification over Semantically Secure Encrypted Relational Data”. In : *IEEE Transactions on Knowledge and Data Engineering* 27.5 (2015), p. 1261-1273. doi : 10.1109/TKDE.2014.2364027.

[39] Wai Kit Wong et al. “Secure KNN Computation on Encrypted Databases”. In : *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’09. Providence, Rhode Island, USA : Association for Computing Machinery, 2009, 139–152. isbn : 9781605585512. doi : 10.1145/1559845.1559862. url : https://doi.org/10.1145/1559845.1559862.

[40] Pascal Paillier. “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes”. In : t. 5. Mai 1999, p. 223-238. isbn : 978-3-540-65889-4. doi : 10.1007/3-540-48910-X_16.

[41] Xiaokui Xiao, Feifei Li et Bin Yao. “Secure Nearest Neighbor Revisited”. In : *Proceedings of the 2013 IEEE International Conference on Data Engineering (ICDE 2013)*. ICDE ’13. USA : IEEE Computer Society, 2013, 733–744. isbn : 9781467349093. doi : 10.1109/ICDE.2013.6544870. url : https://doi.org/10.1109/ICDE.2013.6544870.

[42] Yoshinori Aono et al. “Privacy-preserving deep learning via additively homomorphic encryption”. In : *IEEE Transactions on Information Forensics and Security* 13.5 (2017), p. 1333-1345.

[43] Fanyu Bu et al. “Privacy preserving back-propagation based on BGV on cloud”. In : *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*. IEEE. 2015, p. 1791-1795.

[44] Qingchen Zhang, Laurence T. Yang et Zhikui Chen. “Privacy Preserving Deep Computation Model on Cloud for Big Data Feature Learning”. In : *IEEE Transactions on Computers* 65.5 (2016), p. 1351-1362. doi : 10.1109/TC.2015.2470255.

[45] Qiao Zhang et al. “GELU-Net : A Globally Encrypted, Locally Unencrypted Deep Neural Network for Privacy-Preserved Learning.” In : *IJCAI*. 2018, p. 3933-3939.

[46] Ran Gilad-Bachrach et al. “Cryptonets : Applying neural networks to encrypted data with high throughput and accuracy”. In : *International conference on machine learning*. PMLR. 2016, p. 201-210.

[47] Martin Albrecht, Shi Bai et Léo Ducas. “A Subfield Lattice Attack on Overstretched NTRU Assumptions”. In : *Proceedings, Part I, of the 36th Annual International Cryptology Conference on Advances in Cryptology — CRYPTO 2016 - Volume 9814*. Berlin, Heidelberg : Springer-Verlag, 2016, 153–178. isbn : 9783662530177. doi : 10.1007/978-3-662-53018-4_6. url : https://doi.org/10.1007/978-3-662-53018-4_6.

[48] Mehmood Baryalai, Julian Jang-Jaccard et Dongxi Liu. “Towards privacy-preserving classification in neural networks”. In : *2016 14th Annual Conference on Privacy, Security and Trust (PST)*. 2016, p. 392-399. doi : 10.1109/PST.2016.7906962.

[49] Hervé Chabanne et al. “Privacy-preserving classification on deep neural network”. In : *Cryptology ePrint Archive* (2017).

[50] Ehsan Hesamifard, Hassan Takabi et Mehdi Ghasemi. “Deep neural networks classification over encrypted data”. In : *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*. 2019, p. 97-108.

[51] Qiang Zhu et Xixiang Lv. “2P-DNN : Privacy-preserving deep neural networks based on Homomorphic cryptosystem”. In : *arXiv preprint arXiv :1807.08459* (2018).

[52] Ilaria CHILLOTTI, Marc JOYE et Pascal PAILLIER. "Programmable Bootstrapping Enables Efficient Homomorphic Inference of Deep Neural Networks". In : *Cyber Security Cryptography and Machine Learning*. Sous la dir. de Shlomi DOLEV et al. Cham : Springer International Publishing, 2021, p. 1-19. ISBN : 978-3-030-78086-9.

[53] Xiaoyan LIU et al. "Outsourcing Two-Party Privacy Preserving K-Means Clustering Protocol in Wireless Sensor Networks". In : *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*. 2015, p. 124-133. DOI : 10.1109/MSN.2015.42.

[54] Zoe L. JIANG et al. "Efficient two-party privacy-preserving collaborative k-means clustering protocol supporting both storage and computation outsourcing". In : *Information Sciences* 518 (2020), p. 168-180. ISSN : 0020-0255. DOI : https://doi.org/10.1016/j.ins.2019.12.051. URL : https://www.sciencedirect.com/science/article/pii/S0020025519311624.

[55] Georgios SPATHOULAS, Georgios THEODORIDIS et Georgios-Paraskevas DAMIRIS. "Using homomorphic encryption for privacy-preserving clustering of intrusion detection alerts". In : *International Journal of Information Security* 20 (juin 2021). DOI : 10.1007/s10207-020-00506-7.

[56] Anastasia THEODOULI, Konstantinos A. DRAZIOTIS et Anastasios GOUNARIS. "Implementing private k-means clustering using a LWE-based cryptosystem". In : *2017 IEEE Symposium on Computers and Communications (ISCC)* (2017), p. 88-93.

[57] Zvika BRAKERSKI, Vinod VAIKUNTANATHAN et Craig GENTRY. "Fully homomorphic encryption without bootstrapping". In : *In Innovations in Theoretical Computer Science*. 2012.

[58] Nawal ALMUTAIRI, Frans COENEN et Keith DURES. "K-Means Clustering Using Homomorphic Encryption and an Updatable Distance Matrix : Secure Third Party Data Clustering with Limited Data Owner Interaction". In : *DaWaK*. 2017.

[59] Angela JÄSCHKE et Frederik ARMKNECHT. "Unsupervised Machine Learning on Encrypted Data". In : *IACR Cryptol. ePrint Arch.* 2018 (2018), p. 411.

[60] Georgios SAKELLARIOU et Anastasios GOUNARIS. "Homomorphically Encrypted K-Means on Cloud-Hosted Servers with Low Client-Side Load". In : *Computing* 101.12 (2019), 1813–1836. ISSN : 0010-485X. DOI : 10.1007/s00607-019-00711-w. URL : https://doi.org/10.1007/s00607-019-00711-w.

[61] Zvika BRAKERSKI, Craig GENTRY et Vinod VAIKUNTANATHAN. "(Leveled) Fully Homomorphic Encryption without Bootstrapping". In : *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ITCS '12. Cambridge, Massachusetts : Association for Computing Machinery, 2012, 309–325. ISBN : 9781450311151. DOI : 10.1145/2090236.2090262. URL : https://doi.org/10.1145/2090236.2090262.

[62] Michael ARMBRUST et al. *Above the clouds : A berkeley view of cloud computing*. Rapp. tech. Technical Report UCB/EECS-2009-28, EECS Department, University of California ..., 2009.

[63] Peter M. MELL et Timothy GRANCE. *SP 800-145. The NIST Definition of Cloud Computing*. Rapp. tech. Gaithersburg, MD, USA, 2011.

[64] Hussam ABU-LIBDEH, Lonnie PRINCEHOUSE et Hakim WEATHERSPOON. "RACS : a case for cloud storage diversity". In : *Proceedings of the 1st ACM symposium on Cloud computing*. 2010, p. 229-240.

[65] Yulliwas Ameur et al. “Secure and Non-interactive-NN Classifier Using Symmetric Fully Homomorphic Encryption”. In : *International Conference on Privacy in Statistical Databases*. Springer. 2022, p. 142-154.

[66] Maya Louk et Hyotaek Lim. “Homomorphic encryption in mobile multi cloud computing”. In : *2015 International Conference on Information Networking (ICOIN)*. 2015, p. 493-497. doi : 10.1109/ICOIN.2015.7057954.

[67] Luis Pulido-Gaytan et al. “Privacy-preserving neural networks with Homomorphic encryption : Challenges and opportunities”. In : *Peer-to-Peer Networking and Applications* 14 (mai 2021). doi : 10.1007/s12083-021-01076-8.

[68] Benjamin Fabian, Tatiana Ermakova et Philipp Junghanns. “Collaborative and secure sharing of healthcare data in multi-clouds”. In : *Information Systems* 48 (2015), p. 132-150. issn : 0306-4379. doi : https://doi.org/10.1016/j.is.2014.05.004. url : https://www.sciencedirect.com/science/article/pii/S030643791400088X.

[69] Karim Zkik, Ghizlane Orhanou et Said El Hajji. “Secure scheme on mobile multi cloud computing based on homomorphic encryption”. In : *2016 International Conference on Engineering MIS (ICEMIS)*. 2016, p. 1-6. doi : 10.1109/ICEMIS.2016.7745297.

[70] Craig Gentry. “Fully Homomorphic Encryption Using Ideal Lattices”. In : *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. STOC ’09. Bethesda, MD, USA : Association for Computing Machinery, 2009, 169–178. isbn : 9781605585062. doi : 10.1145/1536414.1536440. url : https://doi.org/10.1145/1536414.1536440.

[71] Adriana López-Alt, Eran Tromer et Vinod Vaikuntanathan. “On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption”. In : *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. 2012, p. 1219-1234.

[72] Hao Chen, Ilaria Chillotti et Yongsoo Song. “Multi-Key Homomorphic Encryption from TFHE”. In : *Advances in Cryptology – ASIACRYPT 2019*. Sous la dir. de Steven D. Galbraith et Shiho Moriai. Cham : Springer International Publishing, 2019, p. 446-472. isbn : 978-3-030-34621-8.

[73] Asma Aloufi et al. “Computing Blindfolded on Data Homomorphically Encrypted under Multiple Keys : A Survey”. In : *ACM Comput. Surv.* 54.9 (2021). issn : 0360-0300. doi : 10.1145/3477139. url : https://doi.org/10.1145/3477139.

[74] Ahmad Al Badawi et al. *OpenFHE : Open-Source Fully Homomorphic Encryption Library*. Cryptology ePrint Archive, Paper 2022/915. https://eprint.iacr.org/2022/915. 2022. url : https://eprint.iacr.org/2022/915.

[75] bessani Alysson Bessani Ricardo Mendes. *”Dependable and Secure Storage in a Cloud-of-Clouds”*. 2016. url : http://cloud-of-clouds.github.io/depsky/.

[76] H. Pussewalage et V. Oleshchuk. “A Patient-Centric Attribute Based Access Control Scheme for Secure Sharing of Personal Health Records Using Cloud Computing”. In : *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*. Los Alamitos, CA, USA : IEEE Computer Society, 2016, p. 46-53. doi : 10.1109/CIC.2016.020. url : https://doi.ieeecomputersociety.org/10.1109/CIC.2016.020.

[77] Yulliwas Ameur, Samia Bouzefrane et Vincent Audigier. “Application of Homomorphic Encryption in Machine Learning”. In : *Emerging Trends in Cybersecurity Applications*. Springer, 2023, p. 391-410.

[78] Florian Bourse et al. “Fast Homomorphic Evaluation of Deep Discretized Neural Networks : 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part III”. In : jan. 2018, p. 483-512. isbn : 978-3-319-96877-3. doi : 10.1007/978-3-319-96878-0_17.

[79] Malika Izabachène, Renaud Sirdey et Martin Zuber. “Practical Fully Homomorphic Encryption for Fully Masked Neural Networks”. In : oct. 2019, p. 24-36. isbn : 978-3-030-31577-1. doi : 10.1007/978-3-030-31578-8_2.

[80] Xiaoyan Liu et al. “Outsourcing two-party privacy preserving k-means clustering protocol in wireless sensor networks”. In : *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*. IEEE. 2015, p. 124-133.

[81] Zoe L Jiang et al. “Efficient two-party privacy-preserving collaborative k-means clustering protocol supporting both storage and computation outsourcing”. In : *Information Sciences* 518 (2020), p. 168-180.

[82] Anastasia Theodouli, Konstantinos A Draziotis et Anastasios Gounaris. “Implementing private k-means clustering using a LWE-based cryptosystem”. In : *2017 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2017, p. 88-93.

[83] Nawal Almutairi, Frans Coenen et Keith Dures. “K-means clustering using homomorphic encryption and an updatable distance matrix : secure third party data clustering with limited data owner interaction”. In : *International Conference on Big Data Analytics and Knowledge Discovery*. Springer. 2017, p. 274-285.

[84] Armknecht Frederik Jäschke et al. Angela. “Unsupervised machine learning on encrypted data”. In : *International Conference on Selected Areas in Cryptography*. Springer. 2018, p. 453-478.

[85] Gounaris Anastasios Sakellariou et al. Georgios. “Homomorphically encrypted k-means on cloud-hosted servers with low client-side load”. In : *Computing* 101.12 (2019), p. 1813-1836.

[86] Michele Minelli. “Fully homomorphic encryption for machine learning”. Thèse de doct. Université Paris sciences et lettres, 2018.

[87] Xiangyun Tang et al. “When Homomorphic Cryptosystem Meets Differential Privacy : Training Machine Learning Classifier with Privacy Protection”. In : *CoRR* abs/1812.02292 (2018). arXiv : 1812.02292. url : http://arxiv.org/abs/1812.02292.

[88] Raphael Kiesel et al. “Potential of Homomorphic Encryption for Cloud Computing Use Cases in Manufacturing”. In : *Journal of Cybersecurity and Privacy* 3.1 (2023), p. 44-60. issn : 2624-800X. doi : 10.3390/jcp3010004. url : https://www.mdpi.com/2624-800X/3/1/4.

[89] Kristin Lauter. “Private AI : Machine Learning on Encrypted Data”. In : *Recent Advances in Industrial and Applied Mathematics*. Sous la dir. de Tomás Chacón Rebollo, Rosa Donat et Inmaculada Higueras. Cham : Springer International Publishing, 2022, p. 97-113. isbn : 978-3-030-86236-7.

[90] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[91] Cynthia Dwork et al. “Calibrating Noise to Sensitivity in Private Data Analysis”. In : *Theory of Cryptography*. Sous la dir. de Shai Halevi et Tal Rabin. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, p. 265-284. isbn : 978-3-540-32732-5.