

I. Introduction :

Plusieurs travaux de préservation de privacy dans k-means sont élaborés sur des distributions de données différentes et en utilisant les méthodes du calcul multi partie sécurisé. Ainsi, la totalité de ces travaux sont démontré dans le modèle semi – honnête où les parties porteuses de données sont considérées comme semi-honnêtes. La manière dont les données sont distribuées sur ces parties (horizontale, verticale ou arbitraire) dicte des changements sur l'algorithme k-means selon le type de cette distribution, ce qui fait varier la manière de préserver la privacy. Sur cette base, nous avons conçu une classification de ces algorithmes, nous permettons ainsi de bien ressortir les besoins réels en privacy tout en montrant leur complexité variée. Dans ce chapitre nous détaillons ces algorithmes et nous analysons selon la distribution des données le niveau de protection des items dans l'algorithme k-means dans le cadre de la préservation de privacy dans cet algorithme [1].

II. L'algorithme de clustering k-means sur un ensemble de données singulier:

Comme déjà vu dans le deuxième chapitre, l'algorithme k-means est souvent exécuté sur un ensemble de données singulier. Nous rappelons dans ce qui suit l'algorithme k-means tout en fixant les données à protéger dans chaque étape de l'algorithme afin de mieux positionner le problème.

Soit D un ensemble de données de n items et p attributs, où g_1, \dots, g_n sont les items et x_{1m}, \dots, x_{pm} sont les valeurs des attributs d'un item g_m (**Fig. 9**).

g_1	x_{11}	x_{12}	x_{1p}
g_2	x_{21}	x_{22}	x_{2p}
.
g_m	x_{m1}	x_{m2}	x_{mp}
.
g_n	x_{n1}	x_{n2}	x_{np}

Fig. 9 L'ensemble de données singulier D

1. L'algorithme k-means [55]:

1. Entrée: k , le nombre de centroïdes.
2. Choix des premiers centroïdes u_1, \dots, u_k
3. Répéter
 Pour chaque point g_i :
 - a. Calculer les distances entre l'item g_i et les différents centroïdes $d_l(g_i, u_l)$.
 /* Chaque item a un vecteur de distances selon les centroïdes */.
 - b. Trouver la distance minimale $\min_l d(g_i, \mu_l)$
 - c. Affecter, l'item g_i au cluster dont le centroïde est le plus proche.Fin pour
 Pour chaque cluster l :
 - a. Calcul du nouveau centroïde: v_l
 - b. $u_l = v_l$Fin pour
4. Jusqu'à aucun changement ne se produit entre les nouveaux et les anciens centroïdes.
5. Sortie: Affectation des items aux clusters finaux.

2. Les entrées de l'algorithme :

- k , le nombre des clusters.
- Les k premiers centroïdes u_1, \dots, u_k , qui sont des items de l'ensemble de données.

3. Les résultats intermédiaires:

- Les distances $d_j(g_i, u_j)$.
- L'affectation intermédiaire aux clusters
- Le nombre des items dans chaque cluster (pour calculer les centroïdes).
- Les centroïdes intermédiaires.

4. Sortie de l'algorithme k-means:

- Affectation finale des items g_i aux clusters.

La préservation de privacy dans l'algorithme k-means dans le modèle semi-honnête implique la protection directe et indirecte des items contre la révélation dans chaque étape de l'algorithme, lorsque les données de ces items sont distribuées sur plusieurs sites ou parties. La protection directe des items est réalisée en empêchant leur révélation au moment de l'exécution de l'algorithme k-means, tandis que la protection indirecte est réalisée en empêchant la révélation des résultats intermédiaires de l'algorithme durant son exécution.

Nous constatons alors que n'importe quel protocole de préservation de privacy dans l'algorithme k-means ne doit révéler que l'affectation finale des clusters aux différentes parties participantes dans le calcul, tout en gardant secret leurs items ainsi que tout les résultats intermédiaires. Cependant la distribution des items implique aussi la distribution des résultats intermédiaires et nous remarquons que pour chaque modèle de distribution de données, il existe un nombre de résultats intermédiaires à protéger, nous avons donc classer les travaux de privacy dans l'algorithme k-means selon le modèle de distribution de données, nous les avons étudié et analysé afin de

ressortir les réels besoins de privacy pour chaque distribution de données et en déduire le meilleur de ces algorithmes pour chaque distribution de données .

III. Approche 1 : L'algorithme de clustering k-means sur un ensemble de données réparti verticalement:

III.1 Positionnement du problème :

Dans un ensemble de données partitionné verticalement chaque item est distribué sur plusieurs parties, de telle sorte que chaque partie possède un nombre de composantes de différents items (**Fig. 10**).

	Partie 1	Partie 2	Partie r
g_1	(x_{11}, x_{12})	(x_{13}, x_{14}, x_{15})	(x_{1p})
g_2	(x_{21}, x_{22})	(x_{23}, x_{24}, x_{25})	(x_{2p})
	(x_{31}, x_{32})	(x_{33}, x_{34}, x_{35})	(x_{3p})
	(x_{41}, x_{42})	(x_{43}, x_{44}, x_{45})	(x_{4p})
g_n	(x_{n1}, x_{n2})	(x_{n3}, x_{n4}, x_{n5})	(x_{np})

Fig. 10 Ensemble de données distribué verticalement sur plusieurs parties

Nous supposons que:

1. L'ensemble de données est réparti sur " r " parties.
2. Le nombre d'items (points, entités) dans l'ensemble de données est " n ".
3. Le nombre de clusters est " k ".
4. " μ_{ij} " est la moyenne du cluster i pour la partie j .
5. Le vecteur $cluster_i$ est le vecteur des items de chaque cluster " i ".

L'algorithme k-means sur un ensemble de données distribué verticalement devient donc:

Initialisation:

- Choix des k premiers centroïdes.

Lors des itérations de k-means:

1. Chaque item a une matrice de distances:
 - Les lignes représentent les clusters
 - Les colonnes représentent les parts des distances sur les différentes parties.
2. Calculer, pour chaque item et d'une façon sécurisée la somme des parts des distances. (protéger les parts de distances de chaque partie)
 - Trouver d'une façon sécurisée le minimum de ces distances (protection des distances).
 - Affectation de chaque item au cluster le plus proche, les parties sachent explicitement le nombre d'items dans chaque cluster.
 - Au moment du calcul des nouveaux centroïdes, les parts locales des centroïdes sont déjà connues aux parties.

Arrêt de l'itération

Dans ce cas, le choix des k premiers centroïdes ne pose pas de problème de privacy tandis que dans les itérations de k-means, le calcul de la somme des parts de distance

des différentes parties implique le besoin de révéler leurs données. L'affectation des items aux clusters les plus proches pose aussi un problème de privacy, lorsqu'on veut trouver le minimum de ces distances pour un item donné. Au moment du calcul des nouveaux centroïdes, les parts locales des centroïdes sont connues explicitement aux différentes parties. De ce fait, le besoin de préserver la privacy provient lors du calcul des distances et l'affectation des items aux clusters les plus proches.

III.2 Etude des travaux de préservation de privacy dans k-means sur des données distribuées verticalement [92][93][95]:

III.2.1 Etude de l'algorithme de J. Vaidya et C.Clifton [92]:

Les auteurs dans cette solution [92] proposent un algorithme k-means dans un scénario où des données personnelles et confidentielles des individus sont réparties sur plusieurs sites et qu'on opte pour leur classification d'une façon globale et non pas locale sans partage de l'information. L'ensemble de données est partitionné verticalement : Les données pour un seul item sont réparties sur plusieurs sites, et chaque site dispose des informations pour toutes les items d'un sous ensemble d'attributs.

Il est supposé que:

- l'existence d'un item dans un site particulier peut être révélée, ce sont les valeurs associées à un item qui sont privées. L'objectif est de grouper un ensemble d'items communs sans révéler aucune des valeurs privées associées à un item.
- Les k moyennes initiales sont choisit arbitrairement.
- Les moyennes μ_i sont semi confidentiels. Chaque site apprend seulement les composantes de μ_i qui correspondent aux attributs dont il tient.
- Toutes les informations sur les attributs des sites sont gardés confidentiels. Chaque site peut indépendamment calculer les composantes de " μ_{ij} " correspondant à ses attributs.
- Le résultat final de l'algorithme pour la partie j est:
 - L'affectation des items aux clusters dans le vecteur *cluster_i*.
 - La valeur/position des moyennes " μ_{ij} " des k clusters.

Nous présentons dans ce qui suit l'algorithme de J. Vaidya et C. Clifton puis nous étudions la procédure de calcul sécurisé du cluster le plus proche qui permet de protéger les distances et les items.

a. L'algorithme:

Entrée: Nombre de parties r , nombre de clusters k et le nombre des items n .

Pour tous les sites $j = 1 \dots r$ faire

Pour tous les clusters $i = 1 \dots k$ faire

 - Initialiser μ_{ij} arbitrairement

FinPour

Fin pour

Répéter

Pour $g = 1 \dots n$ faire (* pour chaque point *)

Pour $j = 1 \dots r$ faire (* pour chaque partie *)

 (* Calcul du vecteur des distances T_j à chaque cluster pour le point g *)

Pour $i = 1 \dots k$ faire

$t_{ij} = |g_j - \mu_{ij}|$

Fin pour

Fin pour

 (* Chaque site met g dans **Cluster**_{cluster_le_plus_proche} *)

cluster_plus_proche()

Fin pour

Pour $j = 1 \dots r$ faire

Pour $i = 1 \dots k$ faire

 (* Affectation des nouvelles moyennes des points des nouveaux clusters **Cluster** _{i} pour les attributs de la partie j *)

$\mu'_{ij} \leftarrow$ moyenne des valeurs des attributs de la partie j pour les points dans le **cluster** _{i}

Fin pour

Fin pour

Jusqu'à **Test_seuil()**

b. La procédure "cluster_plus_proche()":

C'est une procédure qui sert à trouver confidentiellement le cluster le plus proche à un item donné. Cet algorithme est appelé pour chaque item. Chaque partie a comme entrée un vecteur dont les composantes sont les distances de l'item à chacune des moyennes des k clusters. Donc chaque item a une matrice de distances de dimension $k \times r$.

Le but est de trouver la somme des distances locales la plus petite parmi tous les clusters d'une manière sécurisée, sans qu'aucune partie ne connaisse les valeurs de l'autre.

Le problème peut être formellement défini comme suit:

Considérons r parties P_1, P_2, \dots, P_r , chacune avec ses propres k – distances dans le vecteur T_j :

$$P_1 \text{ a } T_1 = \begin{bmatrix} t_{11} \\ t_{21} \\ \vdots \\ t_{k1} \end{bmatrix}, P_2 \text{ a } T_2 = \begin{bmatrix} t_{12} \\ t_{22} \\ \vdots \\ t_{k2} \end{bmatrix}, \dots, P_r \text{ a } T_r = \begin{bmatrix} t_{1r} \\ t_{2r} \\ \vdots \\ t_{kr} \end{bmatrix}$$

O  : t_{ij} correspond    la distance entre une partie de l'item g_i et μ_{ij} , μ_{ij} est la moyenne du cluster i dans la partie j . L'objectif est de calculer l'index ' I ' qui repr  sente le rang de la distance minimale. Formellement trouver:

$$\arg \min_{i=1..k} \left(\sum_{j=1..r} t_{ij} \right)$$

La s  curit   de l'algorithme est bas  e sur trois id  es cl  s:

- D  guiser les composantes distances de chaque site avec des valeurs al  atoires qui vont   tre annul  s lorsqu'ils sont combin  s.
- Comparer les distances de telle sorte que seulement le r  sultat de la comparaison est connu, aucune partie ne sait les distances qui ont   t   compar  s.
- Permuter l'ordre des clusters, donc le r  el sens du r  sultat de la comparaison est inconnu.

L'algorithme n  cessite aussi trois sites non concert  s. Ces sites peuvent   tre parmi les sites des donn  es. Mais ils peuvent   tre aussi externes. Ils ont besoin seulement de savoir le nombre de sites r et le nombre de clusters k . Il est suppos   que les parties non concert  s sont P_1 , P_2 , P_r parmi les sites de donn  es.

b.1 D  guiser les distances (Fig.11):

- Le site P_1 g  n  re un vecteur V_j al  atoire de k valeurs pour chaque site j tel que:

$$\sum_{j=1}^r V_j = 0.$$

- P_1 choisit aussi une permutation π de 1    k .

- P_1 aussi engage chaque site P_j dans l'algorithme de permutation pour g  n  rer la somme du vecteur V_j avec les distances T_j des P_j (Appel de l'algorithme de permutation).

- Le vecteur r  sultant est connu seulement par P_j et il est permut   par π , π est connu seulement par P_1 ; c'est-  -dire P_j a $\pi(V_j + T_j)$, mais il ne conna  t ni π ni V_j

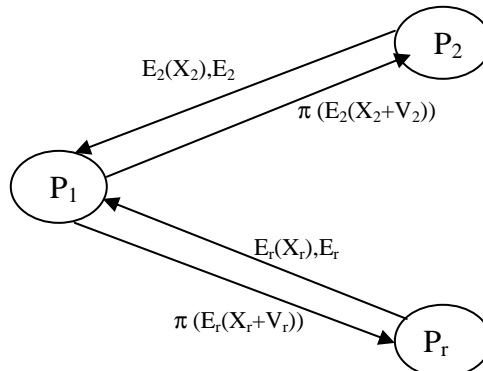


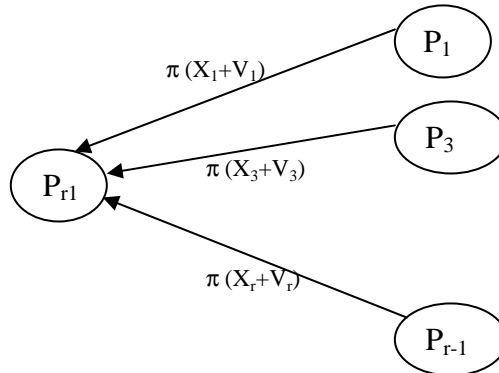
Fig. 11 Etape 1 de la proc  dure cluster plus proche

b.2 Calcul de la distance minimale (Fig. 12):

- P_1 et $P_3 \dots P_{r-1}$ envoient leur vecteur    P_r (sauf P_2)
- Les sites P_2 et P_r s'engagent dans des s  ries d'additions/comparaisons s  curis  es pour trouver l'index (perm  t  ) de la distance minimale.
- Sp  cifiquement, ils veulent trouver si $\sum_{j=1}^r (t_{lj} + v_{lj}) < \sum_{j=1}^r (t_{mj} + v_{mj})$ puisque $\forall l, \sum_{j=1}^r v_{lj} = 0$,

le r  sultat $\sum_{j=1}^r t_{lj} < \sum_{j=1}^r t_{mj}$ montre quel cluster l ou m est le plus proche.

- P_r a toutes les composantes de la somme sauf $T_2 + V_2$.
- P_r g  n  re le vecteur Y qui est la somme des $V_j + T_j$ avec $j = 1, 3, \dots, r$.
- Pour chaque comparaison, il est utilis   le circuit d'  valuation de Yao qui calcule $a_2 + a_r < b_2 + b_r$, sans r  v  ler aucune information sauf le r  sultat de la comparaison.
- Apr  s $(k-1)$ comparaisons, on garde le minimum    chaque fois, l'index du cluster minimal est connu.
- P_2 et P_r connaissent maintenant le cluster minimal dans la permutation minimale. Mais ils ne connaissent pas le r  el cluster    qui, il correspond. Donc, ils envoient le minimum i au site P_1 .
- P_1 diffuse alors le r  sultat $\pi^{-1}(i)$, le cluster appropri   pour l'item.



**Fig. 12 Etape 2 de la proc  dure cluster plus proche
(Calcul de la distance minimale)**

b.3 L'algorithme de la procédure "cluster_plus_proche()" :

Entrée: r parties, chacune avec un vecteur des distances de longueur k . trois de ces parties sont des parties de confiance mais non concertées sont P_1, P_2, P_r

(* **Etape1: entre P_1 et toutes les parties***)

- P_1 génère r vecteurs aléatoires V_j leur somme est à zéro
- P_1 génère une permutation aléatoire π des k éléments.
- Pour $j = 1$ à r faire

$T_i(\text{au } P_j) = \text{som_et_permut}(V_j, \pi(\text{au } P_1), X_j(\text{au } P_j))$

(* cet algorithme est décrit par la suite*)

Fin pour

- P_1 calcule $T_1 = \pi(X_1 + V_1)$

(* **Etape2: entre P_2 et P_r ***)

- Pour $j = 1, 3, \dots, r-1$ faire

P_j envoie T_j à P_r

Fin pour

- P_r calcule $Y = T_1 + \sum_{i=3}^r T_i$

(* **Etape 3: impliquer P_2 et P_r ***)

- $minimal = 1$

- Pour $j = 2..k$ faire

si $\text{som_et_comp_sec}(Y_j + T_{2j} < Y_{\text{minimal}} + T_{2\text{minimal}})$ alors $minimal \leftarrow j$

Fin pour

(* **Etape 4: entre P_r (ou P_2) et P_1 ***)

- Partie P_r envoie $minimal$ à P_1
- P_1 diffuse le résultat $\pi^1(minimal)$

L'algorithme de la procédure "cluster_plus_proche()", utilise à son tour deux sous procédures $\text{som_et_permut}()$ et $\text{som_et_comp_sec}()$ que nous présentons dans ce qui suit.

b.3.1 La procédure som_et_permut():

L'algorithme utilisé est celui de Du et Attallah [87] que nous avons déjà évoqué dans le chapitre III, section VII, il calcule simultanément un vecteur somme, et permute l'ordre des éléments dans le vecteur.

Le problème de permutation est un algorithme de deux parties asymétriques. Il existe deux parties, A et B . B a un vecteur de dimension n , $X = (x_1, \dots, x_n)$ et A a aussi un vecteur de dimension n : $V = (v_1, \dots, v_n)$. A a aussi une permutation π des n nombres. Le but est de donner à B , le résultat $\pi(X+V)$, sans révéler rien d'autre. En particulier, A ne doit pas connaître le vecteur de B et B ne doit pas connaître le vecteur de A , et B ne doit pas connaître π . Selon les propositions, V est un vecteur dont les éléments sont des nombres aléatoires à partir d'une distribution aléatoire et uniforme, utilisé pour cacher la permutation des autres vecteurs.

L'algorithme de permutation comporte les étapes suivantes:

1. B génère la paire de clés (E_k, D_k) pour un schéma de chiffrement homomorphe.
2. B chiffre son vecteur X pour générer le vecteur chiffré $X' = (x'_1, \dots, x'_n)$, $x'_i = E_k(x_i)$.
3. B envoie X' et la clé publique E_k à A .

4. A chiffre son vecteur V en générant le vecteur chiffré $V' = (v'_1, \dots, v'_n)$, $v'_i = E_k(v_i)$.
5. A multiplie les composantes du vecteur X' et V' pour avoir $T' = (t'_1, \dots, t'_n)$, $t'_i = x'_i * v'_i$. selon, la propriété homomorphe du chiffrement,

$$x'_i * v'_i = E_k(x_i) * E_k(v_i) = E_k(x_i + v_i)$$

6. A applique la permutation π au vecteur T' pour avoir le vecteur $T'_p = \pi(T')$ et envoie T'_p à B.
7. B déchiffre les composants de T'_p en donnant le résultat final:

$$T_p = (t_{p1}, \dots, t_{pn}), \quad t_{pi} = x_{pi} + v_{pi}$$

b.3.1 La procédure *som_et_comp_securise()*:

Dans la procédure *som_et_comp_securise()*, Les nombres sont additionnés et comparés en utilisant le circuit d'évaluation de Yao [6] expliqué en détail dans le chapitre III, section V.

c. Communication et complexité arithmétique [92]:

Le coût total de communication dépend du nombre des itérations requises pour converger, qui dépend des données. Supposons que le nombre de parties est r , le nombre d'items est n , et les distances chiffrées peuvent être représentées par m bits.

L'algorithme de permutation nécessite deux cycles de communication. Pour un vecteur de longueur n , le cout total des bits est $2n * m + \text{taille_cle_publique} = O(n)$ bits.

L'algorithme *som_et_comp_securise()* est un protocole biparti, implémenté en utilisant l'évaluation sécurisée d'un circuit de Yao [6]. Il nécessite deux circuits d'addition et un circuit de comparaison. L'addition et la comparaison, les deux nécessitent un nombre de portes linéaire à m . Donc cette étape nécessite un nombre constant de cycles et $O(m)$ bits de communication.

Dans l'algorithme *cluster_plus_proche()*, la communication apparaît dans plusieurs places, le cout de cette communication dépend du nombre d'appel de l'algorithme *som_et_comp_securise()* et l'algorithme de permutation [87] et il est au total de $O(kr)$.

Pour le cout total de l'algorithme, une itération de l'algorithme k-means nécessite un appel du calcul de *cluster_plus_proche()* pour chaque item, donc le cout de communication en bits est de $O(nrk)$.

III.2.2 Etude de l'algorithme de S. Samet et A. Miri [93]:

Le scénario proposé par les auteurs [93] est qu'un ensemble de données est distribué parmi r parties, où chaque partie P_i a l'information sur quelques attributs de toutes les entités de l'ensemble de données. Une procédure de calcul sécurisé du cluster le plus proche est donnée, sans avoir besoin de parties de confiance non concertées. L'ensemble des attributs possédés par une partie P_i est noté $A_i = (a_{i1}, a_{i2}, \dots, a_{iq})$. Pour chaque vecteur centroïde μ_j , P_i a la valeur des composantes correspondant à ces attributs, $(\mu_{j1}, \mu_{j2}, \dots, \mu_{jq})$. Pour calculer la distance à partir d'un centroïde μ_j , chaque partie calcul sa portion qui est:

$$(a_{i1} - \mu_{j1})^2 + (a_{i2} - \mu_{j2})^2 + \dots + (a_{iq} - \mu_{jq})^2$$

Cette valeur est notée d_{ji} . Ainsi la distance d'un item à un centroïde μ_j est:

$$d_{j1} + d_{j2} + \dots + d_{jr} \quad (* \text{ la somme des parts des distances} *)$$

Pour un autre centroïde μ_l la formule est la même:

$$d_{l1} + d_{l2} + \dots + d_{lr}$$

Il faut donc calculer ces deux valeurs pour savoir quel centroïde est le plus proche de l'item. En premier lieu, chaque partie P_i calcule $d_{ji} - d_{li}$ et le note d_i , puis ils utilisent la somme sécurisée [94] citée dans le chapitre III, section VIII.3 pour calculer $\sum_{i=1}^n d_i$. Si le résultat est négatif μ_j est le plus proche, sinon μ_l est le plus proche. Cette étape est répétée pour chaque centroïde avec le prochain jusqu'à ce que le plus proche centroïde soit trouvée.

III.2.3 Etude de l'algorithme de M.C Doganay et T.B Pederson [95]:

Dans cet algorithme, les auteurs [95] reprennent l'algorithme de J. Vaidya et C. Clifton [92] et l'améliorent, la différence principale est l'application du secret partagé additif [89], cité dans le chapitre III, section VIII.1 au lieu du chiffrement homomorphe dans le calcul du cluster le plus proche. Comme dans le premier algorithme, leur solution reposent sur les trois points suivants:

- Chaque partie partage le secret de ses parts de distances avec les autres parties, et leur somme est calculée sur les parts du secret.
- La comparaison des distances est manipulée sur les parts du secret, de telle façon que seulement le résultat de la comparaison est connu.
- L'ordre des clusters est permuté de telle sorte que pour chaque item dans l'ensemble de données, seulement l'index du cluster le plus proche est connu.

Dans la phase 1 du calcul du cluster le plus proche, chaque partie partage le secret de ces parts de distances pour chaque centroïde avec les autres parties. Soit X_{ic} la $i^{\text{ème}}$ part de distance entre l'item à évaluer et le centroïde c . Chaque partie P_i crée un nombre aléatoire α_{ij}^c pour chacune des autres parties, et envoie α_{ij}^c à la partie j pour tous les centroïdes $c \in \{1, \dots, k\}$. La $i^{\text{ème}}$ partie garde $\alpha_{ii}^c = X_{ic} - \sum_{j \neq i} \alpha_{ij}^c$ pour lui-même. A noter que X_{ic} ne peut être calculée à moins que toutes les parties se réunissent pour le recouvrir.

La phase 2, permet de calculer la somme de ces parts de distances, pour chaque cluster c . Soit T_{ic} la $i^{\text{ème}}$ part du secret de la distance entre un item donné et le centroïde c :

$$T_{ic} = \sum_{j=1}^r \alpha_{ji}^c$$

Puisque tous les α_{ij}^c , $i \neq j$, sont des nombres aléatoires, T_{ic} est aussi un nombre aléatoire, donc rien ne peut être dévoilé. Chaque partie, à part la première partie, envoie T_{ic} à la $r^{\text{ème}}$ partie.

Dans la phase 3, seulement les parties 1, 2, 3 et r sont impliquées. La partie r additionne toutes les valeurs T_{ic} pour calculer $D_{rc} = \sum_{i=2}^r T_{ic}$. La partie 1 définit $D_{1c} = T_{1c}$. La distance entre un item donné et le centroïde c est actuellement $D_c = D_{1c} + D_{rc}$.

Puisque la partie 1 n'a pas envoyé T_{1c} , la partie r ne peut pas connaître la distance. La tâche de la partie 1 et r est actuellement de trouver le minimum de la liste (D_1, \dots, D_k) , où la partie 1 connaît D_{1c} , et la partie r connaît D_{rc} pour chaque centroïde. Ceci est réalisé par la comparaison de chaque distance D_c avec la plus petite une par une. Deux problèmes sont soulevés au moment de la comparaison :

1. comment les parties 1 et r peuvent elles comparer les deux nombres : $D_c = D_{1c} + D_{rc}$ et $D_{c'} = D_{1c'} + D_{rc'}$ de telle façon qu'aucune d'elles n'apprendra les valeurs de D_c et $D_{c'}$?
2. comment empêcher les parties 1 et r d'apprendre l'ordre des distances ? Ils doivent connaître seulement le minimum des distances.

Le premier problème est résolu en observant que $D_c < D_{c'}$ si est seulement si $D_{1c} - D_{1c'} < D_{rc} - D_{rc'}$. La partie 1 connaît le coté gauche de l'inégalité, et la partie r connaît son coté droit. Ils peuvent exécuter la comparaison sécurisée en utilisant le protocole de Yao [6]. Pour résoudre le second problème, les parties 1 et r permutent l'ordre des éléments du vecteur (D_1, \dots, D_k) avec l'aide de la partie 2 et la partie 3.

Une fois la phase 3 est terminée, les parties 1 et r peuvent révéler l'index du cluster le plus proche à l'item donné.

Nous présentons dans ce qui suit les sous procédures de la permutation et la comparaison sécurisées.

a. La permutation sécurisée :

Le sous protocole de la permutation sécurisée nécessite l'aide de deux parties, qui sont supposées la partie 2 et 3. La partie 1 envoie ses parts du secret (D_{11}, \dots, D_{1k}) à la partie 2, et la partie r envoie ses parts du secret (D_{r1}, \dots, D_{rk}) à la partie 3. Ensuite, partie 2 et 3 s'accordent sur une permutation et chacune d'elles applique la permutation aux vecteurs qu'ils reçoivent.

b. La comparaison sécurisée:

Dans la dernière étape de la procédure du " plus proche cluster", les parties 1 et r comparent les distances entre l'item courant et tout les centroïdes. Chaque partie possède le secret partagé des k entrées dans le vecteur permuté de ces distances. Pour trouver le minimum, ils manipulent $k-1$ comparaisons avec le minimum courant. Après le calcul de l'élément minimal, la partie 1 informe la partie 2 de l'identité permutée du centroïde le plus proche. Puisque la partie 2 connaît la permutation, elle peut annoncer l'identité réelle du cluster le plus proche à toutes les parties.

La comparaison sécurisée est appliquée en utilisant une modification du protocole de Yao [6] où on utilise la propriété d'homomorphisme du secret partagé vis à vis l'addition et le XOR. Les détails de ce protocole ainsi que la preuve de sa sécurité est dans [96].

c. Communication et complexité arithmétique [95]:

La plupart du coût de communication du protocole est dans la procédure du cluster le plus proche. Dans cette phase, pour chaque item, une partie envoie les parts de du secret de ses portions de distance à chaque partie pour chacun des k clusters. Ceci est

l'équivalent à l'envoi d'un vecteur de longueur k . Chaque entrée de ce vecteur est une part du secret de 32 bits. Puisqu'il y a r parties et chacune d'entre elles envoie un vecteur de secret partagé de longueur k à chacune des autres, le coût de la communication de cette étape est de $32r(r-1)kn$ bits.

Dans la deuxième étape, chaque partie à part la partie 1 envoie un entier de 32 bits à la $r^{\text{ième}}$ partie, donc le coût de la communication est de $32n(r-2)k$ bits.

En troisième phase, seulement le coût de communication de la permutation est considéré, il est appelé à chaque comparaison. Les vecteurs permutés sont de longueur 32λ où λ est un paramètre de sécurité donné par le protocole de comparaison.

III.3. Analyse et critiques [1]:

La première solution proposée par J. Vaidya [92] était pour des données distribuées verticalement, sur plusieurs parties. Dans le protocole proposé, les items de chaque partie sont gardées confidentielles, le calcul sécurisé des distances entre les parties est effectué en introduisant la permutation sécurisé de Du et Attallah [87] qui s'appuient sur les schémas de chiffrement homomorphe de Okamoto et Uchiyama [84] où ils existent de très simples attaques à texte chiffré choisi qui peuvent le recouvrir. La comparaison sécurisée entre les distances est réalisée par le circuit d'évaluation de Yao. Ces primitives sont exécutées pour chaque item de l'ensemble de données ce qui rend le coût de calcul très élevé et impraticable sur de larges ensembles de données. Le protocole nécessite trois parties de confiance non concertées, qui ont plus d'informations que les autres, comme la somme partielle des parts de distances, pour pouvoir appliquer le circuit de Yao dans la comparaison sécurisée et l'ordre de permutation des vecteurs de distances. La démonstration de privacy est donnée dans le modèle semi honnête mais le protocole révèle des informations supplémentaires comme l'affectation intermédiaire des clusters et le nombre d'items dans chaque cluster lors de l'opération de division dans le calcul des centroïdes. Pour éviter le recours à des parties de confiance non concertées et pouvoir appliquer le protocole sur deux parties, S. Samet [93] propose un algorithme de préservation de privacy sur des données verticalement distribuées en utilisant la somme sécurisée [94] dans le calcul de la somme des distances. Mais le protocole ne résout pas le problème de la révélation du nombre des items dans les clusters et les centroïdes intermédiaires. M. C. Doganay et al. [95] ont repris le même schéma de J. Vaidya mais en appliquant le secret partagé additif au lieu du chiffrement homomorphe, l'objectif est de minimiser le coût de calcul et de communication du premier protocole et de démontrer l'efficacité des techniques du secret partagé additif par apport aux crypto systèmes homomorphes pour ces deux paramètres mais en utilisant quatre parties de confiance non concertées au lieu de trois. Ce qui n'améliore pas l'efficacité en terme de préservation de privacy, surtout que l'utilisation des parties de confiance peut violer la privacy si elles deviennent concertées. Nous voyons que les protocoles proposés dans cette approche ne fournissent pas des améliorations originales pour la préservation de privacy, depuis l'algorithme de J. Vaidya [92].

IV. Approche 2 : L'algorithme de clustering *k*-means sur un ensemble de données réparti horizontalement [97][93]:

IV.1 Positionnement du problème [1] :

Dans un ensemble de donn  es partitionn   horizontalement chaque partie poss  de la totalit   de l'item avec toutes les valeurs de ses attributs mais elle n'a qu'un certain nombre d'items de l'ensemble de donn  es (**Fig. 13**).

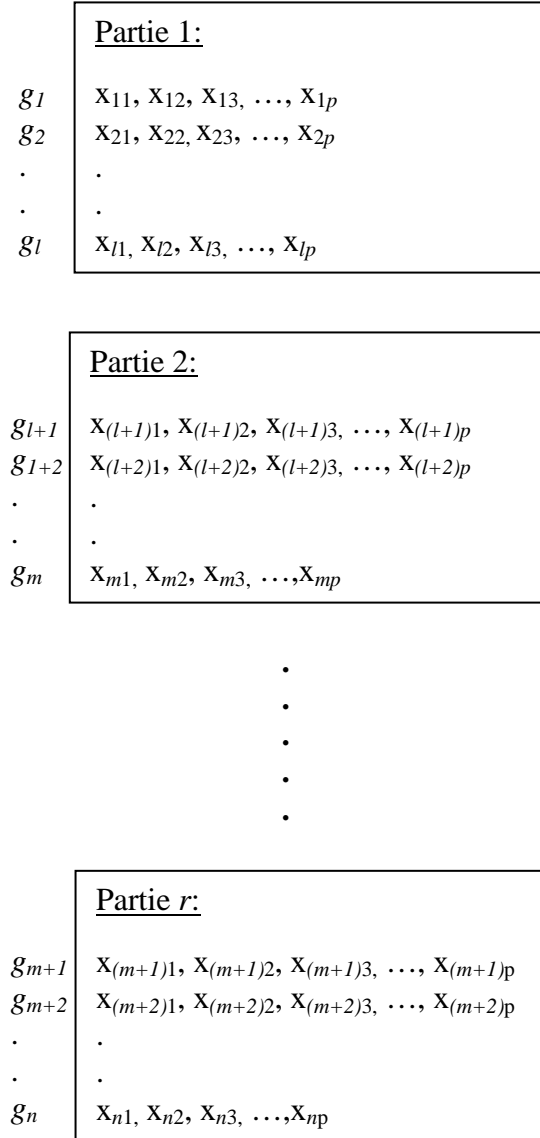


Fig. 13 Ensemble de donn  es distribu   horizontalement sur plusieurs parties

Nous supposons que

1. L'ensemble de donn  es est r  parti sur r parties: not  es P_i ($1 \leq i \leq r$).
2. Le nombre d'items dans l'ensemble de donn  es est n .
3. Le nombre de clusters est k .
4. μ_l est la moyenne du cluster l ($1 \leq l \leq k$).
5. Chaque partie P_i a un sous ensemble de donn  es D_i de D contenant quelques items, tel que $D_i \cap D_j = \emptyset$ et pour tout $1 \leq i, j \leq n \cup D_i = D$

L'algorithme k-means sur un ensemble de donn  es distribu  es horizontalement devient donc:

Initialisation:

- Choix des premiers centroïdes.

Lors des itérations de k-means:

- Une fois les distances sont calculées dans chaque partie, calculer le minimum de ces distances.
- Affectation des items aux clusters.
- Calcul des nouveaux centroïdes, les items d'un même cluster proviennent de parties différentes
 1. Empêcher la révélation du nombre d'items dans chaque cluster ?
 2. Protéger les items au moment du calcul des centroïdes.
 3. Protéger les centroïdes intermédiaires eux même.

Arrêt de l'itération

Nous remarquons que le calcul des distances en lui-même ne viole pas la privacy car chaque partie détient la totalité des composantes d'un item. Le problème se pose lors du calcul des centroïdes des clusters intermédiaires (mis à jour des centroïdes), car les items d'un même cluster peuvent provenir de plusieurs parties d'où l'intérêt de les protéger. Cette étape nécessite aussi la connaissance du nombre d'items dans chaque cluster, ce nombre est une extra information qui ne doit pas être révélée aux différentes parties du protocole. Le protocole de privacy sur des données distribuées horizontalement doit aussi empêcher la révélation d'autres informations supplémentaires comme les centroïdes intermédiaires eux même.

IV.2 Etude des travaux de préservation de privacy dans k-means sur des données distribuées horizontalement :**IV.2.1 Etude de l'algorithme de S. Jha et L. Kruger [97]:**

Dans ce travail, les auteurs [97] présentent deux protocoles de préservation de la privacy dans l'algorithme k-means sur un ensemble de données distribuées sur deux parties P_1 et P_2 seulement. P_1 et P_2 veulent grouper conjointement l'ensemble de données $D_1 \cup D_2$ sans révéler les enregistrements individuels de leurs ensembles de données. Il est présenté alors une version de l'algorithme k-means qui préserve la privacy où seulement les moyennes des clusters sont révélées à la partie P_1 et la partie P_2 dans les différentes étapes de l'algorithme. Le besoin dans le scénario proposé est de préserver la confidentialité des items lors du calcul des centroïdes.

La partie P_1 a l items $\{g_1, \dots, g_l\}$ et la partie P_2 a $n - l$ items $\{g_{l+1}, \dots, g_n\}$. Chaque partie veut grouper conjointement ses items sans révéler n'importe quelle information privée. S'il on suppose qu'il existe une troisième partie de confiance *TTP* (*third trusted party*):

- Les parties P_1 et P_2 exécutent les itérations localement, pour le calcul des distances et l'affectation aux clusters.
- Les nouveaux centroïdes μ_j sont calculés en communiquant avec le TTP:
A chaque itération:
 - Les deux parties ont les $\mu_1, \mu_2, \dots, \mu_k$ envoyé par la TTP.
 - La partie P_1 calcule ses clusters C_{1j}
 - La partie P_2 calcule ses clusters C_{2j}

- P_1 envoie k-pairs $(a_1, b_1), \dots, (a_k, b_k)$ à TTP , où $a_j = \sum_{g_i \in C_{1j}} g_i$ (a_j est la somme des items dans le cluster C_{1j} et b_j est le nombre des échantillons dans le cluster C_{1j}).
- De même, P_2 envoie k-pairs $(d_1, e_1), \dots, (d_k, e_k)$ au TTP .
- Le TTP calcule les k centroïdes (μ_1, \dots, μ_k) et les envoie à P_1 et P_2 , où

$$\mu_j = (a_j + d_j) / (b_j + e_j)$$

Afin de créer une version de préservation de privacy pour l'algorithme k-means sans l'utilisation du TTP , les auteurs proposent un protocole de préservation de privacy pour calculer les centroïdes. A noter que dans les étapes ci-dessus, chaque partie envoie (a_j, b_j) et (d_j, e_j) au TTP et TTP calcule la moyenne: $(a_j + d_j) / (b_j + e_j)$. C'est précisément la fonction pour laquelle le protocole est conçu.

a. Définition du problème de la moyenne pondérée (WAP):

Une partie P_1 a la paire (x, n) , avec x est un nombre réel et n un entier positif. D'une manière similaire, une partie P_2 a la pair (y, m) . Ils veulent conjointement calculer $(x+y)/(n+m)$. En autre mot nous avons besoin d'un protocole de préservation de privacy pour la fonctionnalité suivante: $((x, n)(y, m)) \rightarrow \left(\frac{x+y}{n+m}, \frac{x+y}{n+m} \right)$.

Cette notation veut dire que la première et la seconde partie donne (x, n) et (y, m) en entrée au protocole, et les deux parties reçoivent en sortie $(x+y)/(n+m)$.

Dans l'algorithme de préservation de privacy pour l'algorithme k-means. P_1 et P_2 utilise le protocole P_{WAP} de préservation de privacy pour résoudre WAP au lieu de la troisième partie de confiance TTP pour calculer le centroïde μ_j .

b. L'algorithme:

Entrée: Le nombre de cluster k , les centroïdes initiaux μ_1, \dots, μ_k

Répéter

Pour $i = 1$ à l faire

Classer les l items selon le plus petit proche μ_j :

- Calcul des distances
- Affectation au cluster le plus proche.

Fin pour

Pour $j = 1$ à k faire

- Calculer $a_j = \sum_{g_i \in C_{1j}} g_i$ et $b = |C_{1j}|$

- Calculer μ_i en invoquant le protocole P_{WAP} .

Fin pour

Jusqu'à aucun changement en μ_i

Retourner μ_1, \dots, μ_k

c. Le protocole P_{WAP} :

Deux protocoles sont proposés pour résoudre le problème WAP, le premier protocole est basé sur l'évaluation polynomiale aveuglée (oblivious polynomial

evaluation) [98], et l'autre protocole est basé sur les schémas de chiffrement homomorphe:

c.1 Le protocole P_{WAP} basé sur l'évaluation polynomiale aveuglée:

Le protocole P_{WAP} basé sur l'évaluation polynomiale aveuglée s'appuie sur le protocole P_{PRPE} , ce dernier utilise le protocole de l'évaluation polynomiale aveuglée P_{OPE} [98].

c.1.1 Le protocole P_{PRPE} :

Soit F un ensemble fini, la partie 1 a deux polynômes P et Q avec des coefficients en F . La partie 2 a deux points α et β , les deux parties veulent calculer $P(\alpha)/Q(\beta)$, ils veulent calculer confidentiellement la fonction:

$$((P, Q), (\alpha, \beta)) \rightarrow \left(\frac{P(\alpha)}{Q(\beta)}, \frac{P(\alpha)}{Q(\beta)} \right)$$

Pour résoudre ce problème, on utilise le protocole P_{OPE} : Soit F un corps fini, le problème P_{OPE} peut être défini comme suit: La partie 1 a le polynôme P dans le corps F et la partie 2 a un élément x , après l'exécution du protocole implémentant OPE , la partie 2 doit savoir seulement $P(x)$ et la partie 1 ne doit rien connaître.

Le protocole P_{PRPE} suit alors les étapes suivantes:

1^{ère} étape: Partie 1, choisit un nombre aléatoire $z \in F$, et calcule les deux polynômes zP et zQ , la partie 1 cache les polynômes P et Q .

2^{ème} étape: Partie 2, calcule $zP(\alpha)$ et $zQ(\beta)$ en invoquant le protocole OPE deux fois: $P_{OPE}(zP, \alpha)$ et $P_{OPE}(zQ, \beta)$.

3^{ème} étape: Partie 2 calcule $\frac{P(\alpha)}{Q(\beta)}$ en calculant $\frac{zP(\alpha)}{zQ(\beta)}$ et l'envoie à la partie 1.

c.1.2 Application sur le protocole P_{WAP} :

Dans le protocole P_{WAP} , la partie P_1 a comme entrée (x, n) et la partie P_2 a comme entrée (y, m) respectivement, en invoquant P_{PRPE} , la partie P_1 construit deux polynômes: $P(w) = w * x$ et $Q(w) = w * n$, la partie P_2 pose $\alpha = y$ et $\beta = m$ donc la partie P_2 va calculer $w * (x + y)$ et $w * (n + m)$ en invoquant P_{OPE} , ensuite, elle va calculer: $\frac{w * (x + y)}{w * (n + m)}$.

Dans la deuxième solution, les auteurs utilisent le schéma de chiffrement homomorphe de Benaloh [82] au lieu de l'évaluation polynomiale aveuglée.

c.2 Le protocole P_{WAP} basé sur les schémas de chiffrement homomorphes:

Les parties P_1 et P_2 ont les paires de messages (x, n) et (y, m) . Les deux parties veulent calculer conjointement $\frac{(x+y)}{(n+m)}$ de manière à préserver la privacy. Supposons que la partie P_1 met en place le schéma de chiffrement homomorphe et probabiliste de Benaloh [82] cité dans le chapitre III, section VI.3, et publie les paramètres publics, les étapes du protocole P_H sont :

1^{ère} étape: La partie P_1 chiffre x et n et envoie les valeurs cryptées $x_1 = E(x)$ et $n_1 = E(n)$ à la partie P_2 , où E est la fonction de chiffrement.

2^{ème} étape: La partie P_2 calcule un message aléatoire $z \in M$, et chiffre $z.y$ et $z.m$ pour obtenir $z_1 = E(z.y)$ et $z_2 = E(z.m)$. La partie P_2 calcule ensuite les deux messages suivants et les envoie à la partie P_1 :

$$m_1 = x_1^z * z_1$$

$$m_2 = n_1^z * z_2$$

3^{ème} étape: En utilisant les propriétés du schéma de chiffrement probabiliste, on aura le résultat suivant :

$$m_1 = E(z \cdot x + z \cdot y)$$

$$m_2 = E(z \cdot n + z \cdot m)$$

Donc la partie P_1 peut calculer $z(x+y)$ et $z(n+m)$ et aussi $\frac{(x+y)}{(n+m)}$. La partie P_1 envoie ce résultat à la partie P_2 .

IV.2.2 Etude de l'algorithme de S. Samet et A. Miri [93]:

Pour le calcul de la même fonction, celle de la mis à jour sécurisée des centroïdes, les auteurs dans cet algorithme [93] donne un autre issue qui peut être appliqué pour plusieurs parties, et non pas pour deux parties seulement.

Pour trouver la $j^{ième}$ moyenne, μ_j ($1 \leq j \leq k$), tous les items dans le cluster j sont impliqués. Si l_{ji} est la sommation de tous les items dans la partie P_i et appartiennent au $j^{ième}$ cluster, et f_{ji} le nombre de ces items, donc le nouveau centroïde est :

$$\mu_j = \frac{\sum_{i=1}^n l_{ji}}{\sum_{i=1}^n f_{ji}} \quad (1)$$

Cette somme doit être calculée entre les parties d'une manière sécurisée. On considère le cas d'un seul centroïde, il existe r parties où chacune possède les deux valeurs x_i et y_i , et ils désirent calculer d'une façon sécurisée:

$$\frac{\sum_{i=1}^r x_i}{\sum_{i=1}^r y_i}$$

En premier lieu en utilisant l'addition multi partie sécurisée développée par [99], les parties calculent séparément les e_i et les s_i tel que:

$$\sum_{i=1}^r x_i = \prod_{i=1}^r e_i, \quad \sum_{i=1}^r y_i = \prod_{i=1}^r s_i$$

Ensuite, une partie, par exemple P_1 reçoit $t_i = \frac{r_i}{s_i} (2 \leq i \leq r)$ de toutes les autres parties, et calcule, qui est égal à l'expression (1).

IV.3 Analyse et critiques:

S. Jha et al. [97] ont proposé deux protocoles de préservation de privacy dans l'algorithme k-means sur deux parties seulement. Dans le scénario indiqué, seulement les items de chaque partie sont gardés confidentiels, les centroïdes intermédiaires sont révélés aux deux parties et le calcul des distances se fait localement dans chaque partie. Les primitives de sécurité sont utilisées pendant le calcul des moyennes (centroïdes) pour préserver la confidentialité des items de chaque partie. Le premier protocole (nommé OPE) est basé sur l'évaluation polynomiale aveuglée (oblivious polynomial evaluation) donné par Naor et Pinkas [98]. Le deuxième protocole (nommé DPE) est basé sur le schéma de chiffrement homomorphe. La solution proposée suppose toujours que les deux parties sont semi – honnêtes, le but était d'évaluer les deux protocoles expérimentalement. Le schéma de chiffrement homomorphe donne plus d'efficacité que l'OPE pour les deux paramètres coût de calcul et de communication, mais leur solution ne peut pas être étendu à plusieurs parties, et les deux protocoles révèle des extra information comme la révélation des centroïdes intermédiaires et l'affectation des clusters aux deux parties. S. Samet et al. [93] ont proposé un protocole qui s'appuie sur une méthode de division sécurisée qui protège les items de chaque partie et empêche la révélation du nombre d'items dans chaque cluster. Le protocole est applicable dans un environnement multi-partie, mais les centroïdes intermédiaires sont toujours révélés. Deux protocoles sont proposés dans cette approche et le protocole de S. Samet [93] est vus comme mieux que celui de S. Jha du moment où il cache le nombre des items dans chaque cluster lors de la mise à jour des centroïdes.

V. Approche 3 : Algorithme de clustering k-means sur un ensemble de données réparti arbitrairement [32][91][93][94]:

V.1 Positionnement du problème :

Bien que la distribution arbitraire de données ne soit pas pratique, l'intérêt de considérer un tel partitionnement est que les protocoles dans ce modèle peuvent être appliqués sur les données horizontalement et verticalement distribuées. Chaque partie, possède tout ou quelques attributs de tous ou quelques items de l'ensemble de données (Fig. 14).

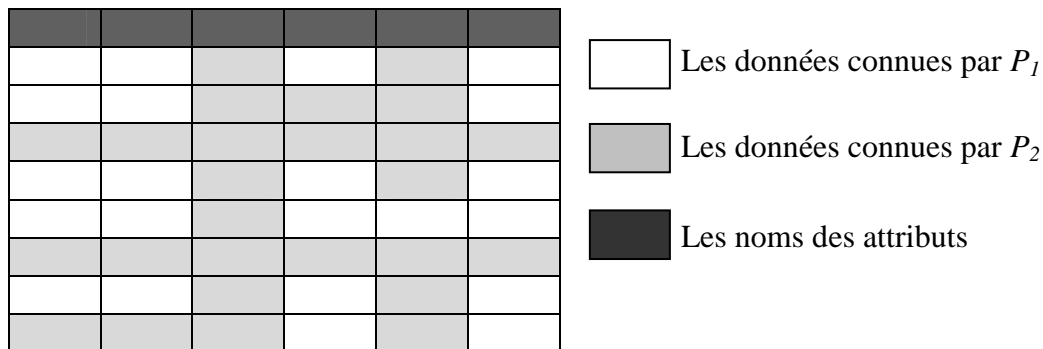


Fig. 14 Ensemble de données distribué arbitrairement sur deux parties

Le besoin de privacy dans l'algorithme k-means dans une distribution de données arbitraire regroupe tout les cas déjà vu dans la distribution horizontale et verticale. Les outils de sécurité sont introduits dans les étapes : Affectation au cluster le plus proche et mise à jour des centroïdes :

Initialisation:

- Choix des premiers centroïdes.

Lors des itérations de k-means:

- Pour chaque item: Affectation au cluster le plus proche.
 - Calculer les parts de distances par à apport à chaque centroïde, localement dans la partie (protection des centroïdes intermédiaires) .
 - Pour chaque centroïde (cluster), faire la somme des parts de distances de chaque item d'une façon sécurisée (protection des distances).
 - Trouver le minimum de ces distances (protection des distances).
 - Affectation des items aux clusters intermédiaires.
- Pour chaque centroïde: Calcul des nouveaux centroïdes:
 - les items d'un même cluster proviennent de parties différentes et pour un item, une partie ne possède pas tous les attributs (protection des items).
 - Protection du nombre des items dans chaque cluster.

Arrêt de l'itération.

V.2 Etude des travaux de préservation de privacy dans k-means sur des données distribuées arbitrairement :

V.2.1 Etude de l'algorithme de G. Jagannathan et R.N. Wright [46]:

Dans ce travail les auteurs [46] introduisent pour la première fois la notion de données arbitrairement partitionnées. Ils proposent un algorithme k-means qui préserve la privacy dans le cas d'un ensemble de données réparti sur deux parties P_1 et P_2 seulement. Dans ce cas différents attributs pour différents items peuvent être possédés par chaque partie.

a. Le scénario proposé :

- Chaque item g_i est partitionné en deux sous ensembles d'attributs disjoints: g_i^1 et g_i^2 de telle sorte que g_i^1 connu par P_1 et g_i^2 connu par P_2 .
- L'ensemble des attributs dont les valeurs connus par P_1 pour un item g_i peut être ne pas égal à l'ensemble des attributs dont les valeurs sont connus par pour P_2 un autre item g_j ($i \neq j$).
- Il est possible que $g_i^1 = \emptyset$ ou $g_i^1 = g_i$. Ainsi, un item donné peut être complètement possédé par P_1 ou par P_2 .
- Si un objet est complètement possédé par P_1 (respectivement P_2) son existence est inconnue pour P_2 (respectivement P_1).
- Les deux parties connaissent le nombre total n des items ou ils connaissent le rang des valeurs pour tous les attributs.
- Le résultat final de l'algorithme est l'affectation d'un numéro d'un cluster entre 1 et k pour chaque item. Si un item est partagé, alors les deux parties doivent savoir l'affectation de cet item. Sinon, seulement la partie qui possède l'item reçoit l'affectation.

b. L'algorithme :

Soit μ_j^1 pour $1 \leq j \leq k$, la part de P_1 pour le $j^{\text{ième}}$ centroïde et μ_j^2 pour $1 \leq j \leq k$ la part de P_2 pour le même cluster. Les centroïdes μ_j des clusters sont donnés par $\mu_j^1 + \mu_j^2$ pour $1 \leq j \leq k$. Pour chaque item g_i , P_1 et P_2 calculent les distances $d(g_i, \mu_j)$ pour $1 \leq j \leq k$, entre l'item et chacun des k centroïdes.

b.1 Le protocole sécurisé de calcul du cluster le plus proche :

Considérons l'item $g_i = (x_{i1}, \dots, x_{ip})$. Cet item peut être possédé par P_1 et P_2 ou partagé entre les deux. Supposons le cas où l'item est partagé entre P_1 et P_2 . Soit $x_{ip_1}, \dots, x_{ip_s}$ les coordonnées de g_i appartenant à P_1 et le reste $(p-s)$ les coordonnées appartenant à P_2 . Pour calculer le cluster le plus proche, le protocole doit initialement calculer la distance entre l'item g_i et chacun des k centroïdes. Pour chaque cluster $1 \leq j \leq k$, le calcul de la distance $d(g_i, \mu_j)$ est :

$$(d(g_i, \mu_j))^2 = (x_{i1} - \mu_{j1})^2 + (x_{i2} - \mu_{j2})^2 + \dots + (x_{il} - \mu_{jl})^2$$

Puisque $\mu_{jt} = \mu_{jt}^1 + \mu_{jt}^2$ où μ_{jt}^1 est la part de P_1 et μ_{jt}^2 est la part de P_2 du centroïde

$$\mu_{jt} : (d(g_i, \mu_j))^2 = (x_{i1} - (\mu_{j1}^1 + \mu_{j1}^2))^2 + \dots + (x_{ip} - (\mu_{jp}^1 + \mu_{jp}^2))^2$$

$$(d(g_i, \mu_j))^2 = \sum_{m=1}^p x_{im}^2 + \sum_{m=1}^p (\mu_{jm}^{(1)})^2 + \sum_{m=1}^p (\mu_{jm}^{(2)})^2 + 2 \sum_{m=1}^p \mu_{jm}^{(1)} \mu_{jm}^{(2)} - 2 \sum_{m=1}^p \mu_{jm}^{(1)} x_{im} - 2 \sum_{m=1}^p \mu_{jm}^{(2)} x_{im}$$

Pour calculer confidentiellement le terme $\sum_{m=1}^p x_{im}^2$, la partie P_1 calcule la somme qui implique $x_{ip_1}, \dots, x_{ip_s}$ et la partie P_2 calcule le reste de la somme des coordonnées, le second et le troisième terme sont calculés chacun séparément par les deux parties P_1 et P_2 . Ensuite les deux parties utilisent le produit scalaire sécurisé [88], cité dans le chapitre

III, section VII, pour calculer les parts al  atoires des trois derniers termes, le principe des parts al  atoires est aussi expliqu   dans le chapitre III, section VIII.2. Puisqu'une somme impliquant les parts al  atoires a comme r  sultat des parts al  atoires alors: $(d(g_i, \mu_j))^2 = \alpha_{ij} + \beta_{ij}$ o   α_{ij} est la part al  atoire connue par P_1 et β_{ij} est la part al  atoire connue par P_2 . P_1 a un vecteur de longueur k :

$A = (\alpha_{i1}, \dots, \alpha_{ik})$ et P_2 a le vecteur $B = (\beta_{i1}, \dots, \beta_{ik})$ et ils calculent confidentiellement l'index j tel que $\alpha_{ij} + \beta_{ij}$ est le minimum, en utilisant le circuit d'  valuation de Yao [6].

b.2 Le protocole s  curis   de mis    jour des centro  des:

Dans l'  tape suivante, les deux parties P_1 et P_2 recalculent les nouveaux centro  des. Supposons que la partie P_1 a les items $g_{i_1}^{(1)}, \dots, g_{i_l}^{(1)}$ et la partie P_2 $g_{i_1}^{(2)}, \dots, g_{i_m}^{(2)}$ pour chaque cluster $1 \leq i \leq k$, chacun des $g_{ij}^{(2)}$ et $g_{ij}^{(1)}$ est un p -uplet et $g_{ij}^{(1)}$ d  note la $j^{\text{i  me}}$ coordonn  e du p -tuple correspondant.

La partie P_1 calcule les parts s_j et n_j pour $1 \leq j \leq p$, o   $s_j = \sum_{e=1}^l g_{i_e j}^{(1)}$ et n_j d  note le nombre des items dans le cluster i pur la partie P_1 .

Similairement, la partie P_2 calcule les parts t_j et m_j pour $1 \leq j \leq p$, o   $t_j = \sum_{e=1}^m g_{i_e j}^{(2)}$ et m_j d  note le nombre des items dans le cluster i pour la partie P_2 . La $j^{\text{i  me}}$ coordonn  e de l' $i^{\text{i  me}}$ centro  de est donn  e par : $\mu_{ij} = (s_j + t_j)/(n_j + m_j)$. Ceci est une fonction    quatre entr  es, qui peut   tre calcul  e par le circuit d'  valuation de Yao [6]. Le protocole de Bar-Ilan et Beaver [100] peut   tre utilis   pour calculer l'inverse $(n_j + m_j)^{-1}$ comme $u+v$ o   P_1 conna  t u et P_2 conna  t v . Un protocole de multiplication peut ensuite   tre utilis   pour calculer $a+b = (s_j + t_j)(u+v)$ o   P_1 conna  t a et P_2 conna  t b .

c. Communication et complexit   arithm  tique [46]:

Dans le protocole s  curis   de calcul du cluster le plus proche, on calcule pour chaque item g_i , le cluster le plus proche o   chaque item a p composantes. Supposons que chaque composante est repr  sent  e par c bits. La communication intervient lorsque les deux parties s'engagent dans le protocole scalaire s  curis   pour calculer la distance entre chaque item g_i et le centro  de μ_j . La complexit   de communication de chaque produit scalaire est $O(cl)$. Donc il consomme un cout de communication de $O(kcl)$ bits pour calculer la distance entre l'item g_i et tout les k centro  des. Le co  t du circuit d'  valuation de Yao a besoin d'une communication de $O(c)$ bits par comparaison et $O(lc)$ bits pour calculer la composante minimale. La complexit   totale de l'invocation du protocole s  curis   du cluster le plus proche est $O(kcl)$.

Dans le protocole s  curis   de mis    jour des centro  des, chaque composante du centro  de n  cessite une communication de $O(c)$ bits. Donc la complexit   de communication du recalcul du centro  de est $O(kcl)$.

Ainsi, la complexit   totale de communication de l'algorithme de pr  servation de privacy pour l'algorithme k-means est $O(nckl)$ par it  ration .

V.2.2 Etude de l'algorithme de C. Su et F. Bao [101]:

Dans un ensemble de données, si les valeurs des variables sont dans des attributs différents, alors, il est probable que quelques variables prennent des valeurs très grandes. Les variables avec une grande variabilité vont dominer la métrique. Les auteurs, dans ce travail passent par une normalisation sécurisée de l'ensemble de données avant d'exécuter l'algorithme k-means afin de forcer les attributs à avoir une gamme de valeurs communes. Soit \bar{x} et σ la moyenne et l'écart type d'un item à p -attributs x_i . Dans l'ensemble de données, on peut faire la normalisation et avoir des items normalisés $x'_i = \frac{x_i - \bar{x}}{\sigma}$. Après la normalisation, l'algorithme de clustering k-means peut démarrer. L'algorithme de Su [101] comporte trois sous protocoles sécurisés, un pour la normalisation sécurisée des données et les deux autres pour calculer le cluster le plus proche et la mis à jour des centroïdes. Le scénario proposé est le même que celui dans l'algorithme précédent.

a. Le protocole de la normalisation sécurisée :

Ce protocole résous le problème du calcul privé de la moyenne et de l'écart type. Supposons que la partie P_1 a n_1 items et la partie P_2 a n_2 items. La somme des items de la partie P_1 est $S^{(1)} = \sum_{i=1}^{n_1} g_{i=1}^{n_1}$ et la somme des items de la partie P_2 est $S^{(2)} = \sum_{i=1}^{n_2} g_{i=1}^{n_2}$. La

moyenne M est calculée comme : $M = \frac{S^{(1)} + S^{(2)}}{n_1 + n_2}$, et l'écart type peut être calculé :

$$\sigma = \sqrt{\frac{1}{n_1 + n_2} \left(\sum_{i=1}^{n_1} (g_i^{(1)} - M)^2 + \sum_{i=1}^{n_2} (g_i^{(2)} - M)^2 \right)}$$

Dans ce cas il faut calculer la moyenne M d'une façon sécurisée. Ceci est l'objet du sous protocole de la mis à jour des centroïdes.

b. Le protocole du calcul du cluster le plus proche :

Dans cette étape, les auteurs reprennent le même protocole que l'algorithme précédant pour le calcul des distances entre un item et les centroïdes des différents clusters en utilisant le produit scalaire sécurisé. Mais ils n'indiquent rien sur la façon dont ils comparent les distances pour trouver la distance minimale.

c. Le protocole de mis à jour des centroïdes :

Dans cette étape, les auteurs proposent un autre issue pour calculer d'une manière sécurisée le terme : $\mu_{ij} = (s_j + t_j)/(n_j + m_j)$ le nouveau centroïde pour le cluster j . L'idée est d'utiliser le protocole privé 2^l -approximation sur le corps fini F_p proposé par Kiltz et al. [102] et aide avec l'algorithme de l'évaluation polynomiale aveuglée (oblivious polynomial evaluation) [98], basé sur le chiffrement homomorphe du cryptosystème de Paillier [85], cité dans le chapitre III, section VI.4.

e. Communication et complexité algorithmique :

Pour le calcul sécurisé des distances la complexité est de $O(knm)$. Pour les mis à jour des centroïdes, l'algorithme spécifié s'exécute en un nombre constant de cycles de communication entre les deux parties. La complexité de l'algorithme dépend aussi de l'exactitude du résultat demandé.

V.2.3 Etude de l'algorithme de P. Bunn et R. Ostrovsky [103]:

La solution proposée, décrit un protocole biparti pour l'algorithme de clustering k-means, qui garantit la privacy, la contribution principale est de calculer efficacement les itérations multiples k-means sans révéler les valeurs intermédiaires. Les auteurs dans [103] proposent un sous protocole sécurisé pour chaque opération de l'algorithme k-means. Les sous protocoles utilisés sont à base du protocole du produit scalaire sécurisé [88] :

- Le protocole "plus grand que N" (Bigger Then N Protocol) BTPNB: Ce protocole détermine si la somme des valeurs des deux parties P_1 et P_2 est plus grande que N .
- Le protocole binaire (To Binary Protocol) TBP: P_1 et P_2 ont les parts de d'une valeur $X \in \mathbb{Z}_N$. Si $X = x_K \dots x_1$ est l'expansion binaire de X . Alors ce protocole retourne les parts de x_i , pour chaque $1 \leq i \leq K$. En un autre mot, $x_i = x_i^{(1)} + x_i^{(2)} \pmod{N}$.
- Le protocole du minimum entre deux nombres (Find Minimum of 2 numbers Protocol) FM2NP: P_1 et P_2 partagent deux nombres. Ce protocole retourne les parts de la position du nombre le plus petit (0 ou 1).
- Le protocole du minimum entre k nombres (Find Minimum of k numbers Protocol) FMkNP: c'est une extension du protocole FM2NP, mais cette fois ci P_1 et P_2 reçoivent les parts du vecteur $(0, \dots, 1, \dots, 0)$ comme résultat, où 1 apparaît dans la $m^{ième}$ coordonnée si le $m^{ième}$ est le plus petit.
- Le protocole de Distance (Distance Protocol) DistP: calcule la distance partagée entre deux vecteurs dans \mathbb{Z}_N^p (invoque le protocole SPP).
- Le protocole de Division (Division Protocol) DivP: Calcule le quotient d'un dividende partagé sur un diviseur partagé.

a. L'algorithme :**a.1 Le protocole du cluster le plus proche :**

Pour chaque item $g_i \in D$, P_1 et P_2 répètent les étapes suivantes :

1. Trouver les distances carrées à chaque centroïde :

Puisque P_2 a les parts aléatoires des items de P_1 et les centroïdes, P_2 peut passer par le calcul de **DistP**, pour obtenir pour chaque centroïde du cluster j , la distance chiffrée $d_{i,j}$ de l'item g_i au centroïde du cluster j . P_2 désordonne chaque

vecteur de distance et le retourne à P_2 , de sorte que chaque cluster j , P_1 et P_2 partagent le vecteur $d_i = (d_{i1}, \dots, d_{ik})$ (* d_{ij} est partagé entre P_1 et P_2 *).

2. Comparer les distances :

P_1 et P_2 exécutent le protocole **FMkNP** sur $d_i^{(1)}$ et $d_i^{(2)}$ pour obtenir une part de la représentation vectorielle de la location du cluster le plus proche à D_i :

$$C_i = (0, \dots, 0, 1, 0, \dots, 0)$$

A noter que, C_i est partagé entre P_1 et P_2 : $C_i = C_i^{(1)} + C_i^{(2)}$ et P_1 chiffre sa part et l'envoie à P_2 , où '1' apparaît dans la coordonnée j si le centre du cluster μ_j est le plus proche à g_i .

a.2 Calcul des nouveaux centroïdes :

Le suivant va être donné pour chaque cluster $1 \leq j \leq k$. Le calcul est partagé en trois étapes :

Etape1: P_2 va calculer la somme des points de données dans le cluster j .

Etape2: P_2 va calculer le total du nombre de points dans le cluster j .

Etape3: Le résultat de l'étape 1 va être divisé sur le résultat de l'étape 2.

Pour simplifier la notation, $E(C_i)$, veut dire $(E(C_{i1}), \dots, E(C_{ip}))$.

1. Calcul de la somme des items dans le cluster j : dans cette étape, P_2 va calculer la somme des items dans le cluster j , on note cette somme comme :

$$S_j \in Z_N^p = \sum_{i=1}^n \begin{cases} g_i, & \text{si } g_i \in \text{cluster } j \\ 0, & \text{sinon} \end{cases}$$

A la fin de cette étape, P_1 et P_2 vont partager $S_j = S_j^{(1)} + S_j^{(2)}$ (ici l'addition est dans Z_N^p). Appelé à partir de l'étape 1 en dessus que pour chaque point g_i , P_2 a $E(C_i^{(1)})$ et $C_i^{(2)}$, où :

$$C_i^{(1)} + C_i^{(2)} = C_i = (0, \dots, 0, 1, \dots, 0..0)$$

Où '1' apparaît dans le $m^{\text{ième}}$ cluster si g_i est proche du cluster m . En plus, le cluster j va totaliser :

$$S_j = \sum_{i=1}^n C_{ij} g_i$$

En utilisant les propriétés homomorphes, P_2 peut calculer (un chiffrement) de S_j , en retournant une part randomisée à P_1 de telle sorte qu'ils partagent S_j comme désiré.

2. Le nombre des items dans le cluster j : Maintenant P_1 et P_2 souhaitent partager le nombre total des points dans le cluster j , noté par T_j . A noter que :

$$T_j = \sum_{i=1}^n C_{ij}$$

i.e T_j peut être trouvé en additionnant la j -ème coordonnée de C_i pour chaque i . P_2 peut calculer T_j en utilisant ces propres parts de C_i et les parts cryptés de P_1 et randomiser son calcul. P_1 et P_2 partagent T_j .

3. Le centro  de des items dans le cluster j : Dans cette   tape P_1 et P_2 veulent diviser $S_j^A + S_j^B$ (   partir de l'  tape 1) par le nombre total des points de donn  es T_j dans le cluster j pour obtenir le nouveau centro  de v_j :

$$v_j = (S_j^{(1)} + S_j^{(2)}) / (T_j^{(1)} + T_j^{(2)})$$

P_1 et P_2 ex  cutent le protocole **DivP** p fois (une fois pour chaque composante) sur les entr  es $P = S_{jl}^{(1)} + S_{jl}^{(2)}$ (le l -i  me coordonn  e de S_j) et le diviseur $D = T_j^{(1)} + T_j^{(2)}$ (   noter que n  cessairement $D \in [1..n]$).

b. Description du sous protocole **DivP** :

1. Utilisation du protocole γ -calcul :

Entr  e : P_1 et P_2 partagent $g \in \mathbb{Z}_N$

Sortie : P_1 et P_2 partagent $\gamma \in \mathbb{Z}^{k-1}_2$, o   la i -i  me coordonn  e de γ est 1 si et seulement si $2^i D < N$.

Cout : Le co  t de la communication de ce protocole est $O(k\xi_s)$, o   ξ_s est le co  t du protocole s  curis   **FMN2P**.

1. P_1 et P_2 ex  cutent **FMN2P** $(k-1)$ fois : la i -i  me fois sur $(N-2^{i-1}D, N-2^{i-1}D-1)$, dont les sorties sont les parts de O_i (   noter qu'ils ex  cutent la version modifi  e du protocole, dans le cas de l'  galit   le protocole fait toujours sortir '0').
2. Soit $O = (O_1, \dots, O_{k-1})$ et notons que $O = (1, \dots, 1, 0, *, \dots, *)$ o   le premier '0' appara  t dans la i -i  me coordonn  e, si ' i ' est    la premi  re fois $2^i D \geq N$. P_1 et P_2 peuvent modifier ceci pour partager $\gamma = (1, \dots, 1, 0, \dots, 0)$ en ex  cutant SPP $(k-1)$ fois, sur la i -i  me fois, ils affectent $\gamma_{i+1} = \gamma_i O_{i+1}$

2. Le protocole **DivP** :

1. Ex  cuter le protocole γ -calcul, P_1 et P_2 calculent et partagent $\gamma \in \mathbb{Z}^{k-1}_2$, o   la i -i  me coordonn  e de γ est 1 si et seulement si $2^i D < N$.
2. Pour chaque $1 \leq i \leq k$, P_1 et P_2 ex  cutent **FMN2P** sur $(P_i, P_i - \gamma_{k-i} 2^{K-i} D)$, o   :
 - O_i d  note la sortie    l' i -i  me appel de **FMN2P**.
 - $P_i = P_{i-1} - O_{i-1} 2^{K-i-1} D$
 - $P_1 = P$ et $\gamma_0 = 1$.
3.    noter que $Q = \sum_{i=1}^K O_i 2^{k-i}$, qui peut   tre localement calcul   par P_1 et P_2    partir de leurs parts de O_i    partir de l'  tape pr  c  dente.

Le co  t de communication de l'algorithm   de P. Bunn et R. Ostrovsky est lin  aire au nombre des items, le nombre de parties, le nombre d'attributs de chaque item, le nombre de clusters et le nombre d'it  rations dans l'algorithm   k-means, puisque on s  curise chaque op  ration de calcul dans l'algorithm   k-means. Ainsi que toutes ces op  rations ont   t   s  curis  es en utilisant le produit scalaire s  curis   [88], qui lui-m  me utilise les crypto syst  mes de Paillier [85].

V.2.4 Etude de l'algorithme de J. Sakuma et S. Kobayashi [104]:

Dans ce nouveau protocole [104], les auteurs donnent un algorithme de préservation de privacy pour l'algorithme k-means original, dans la mesure où ils introduisent la notion de « user-centric privacy », où les parties sont de simples utilisateurs, ce qui implique un grand nombre de parties, l'accent est mis sur la scalabilité et le traitement des environnements incertains des réseaux comme l'asynchronisme et les pannes.

a. L'algorithme :

Soit $Z = \{z_{ij}\}$, $z_{ij} \in \{0,1\}$ un ensemble d'étiquettes. Si la donnée x_i appartient au $i^{\text{ème}}$ cluster, l'étiquette du cluster $z_{ij} = 1$ sinon $z_{ij} = 0$. Les étiquettes des clusters et les centroïdes sont mis à jour alternativement et répétitivement jusqu'à la convergence comme suit :

$$\mu_j = \frac{\sum_{i=1}^n z_{ij} x_i}{\sum_{i=1}^n z_{ij}} \quad (\text{Mis à jour des centroïdes})$$

$$z_{ij} = \begin{cases} 1 & \text{si } i = \arg \min_k (x_j - \mu_k)^T (x_j - \mu_k) \text{ (mettre à jour l'étiquette du cluster)} \\ 0 & \text{sinon} \end{cases}$$

Supposons qu'il existe n parties P_i ($i=1, \dots, r$). La partie P_i possède un sous ensemble de données D_i comme l'entrée du protocole où $\bigcup_i D_i = D$. Après l'exécution du protocole, la partie P_j apprend un ensemble d'étiquettes du cluster correspondant aux items $g_j \in D_i$ et rien d'autre.

a.1 Le protocole du cluster le plus proche :

Basé sur l'étiquetage des clusters, les auteurs proposent le protocole "Private Nearest Cluster center Determination (NCD)" il calcule confidentiellement et d'une façon décentralisé les étiquettes des clusters les plus proches. La confidentialité est préservée à ce niveau en utilisant les parts aléatoires et la comparaison confidentielle basée sur le circuit sécurisé d'évaluation de Yao [6]. Après son exécution, chaque partie apprend un ensemble d'étiquettes des clusters correspondants à leurs propres données et rien d'autre. La complexité du protocole proposé est $O(\log n)$, avec n est le nombre de parties, le calcul est décentralisé dans chaque nœud et d'une façon asynchrone et insensible aux pannes. Ce qui n'est pas le cas dans les protocoles déjà proposés. Les auteurs aussi, ont évalué expérimentalement leur protocole.

a.2 Le protocole de mis à jour des centroïdes:

Utilisé pour calculer confidentiellement les centroïdes le protocole "Private Asynchronous average computation (AAC)", est une extension cryptographique du protocole newscast proposé par Kowalsczyk et al. [105] qui calcule la moyenne des valeurs distribuées à travers des réseaux P2P sans transférer les données dans un dépositaire central. A ce niveau la confidentialité est préservée en utilisant les crypto systèmes homomorphes à clé publique de Paillier [85].

V.3 Analyse et critiques [1]:

L'idée d'un protocole de protection de privacy sur des données arbitrairement partitionné a été introduite par G. Jagannathan et R. N Wright [46], leur solution préserve la privacy dans l'algorithme k-means entre deux parties. Aucune autre partie de confiance n'est utilisée. L'idée est que toutes les valeurs calculées dans les étapes intermédiaires et qui doivent être inconnues sont partagées en des valeurs aléatoires (principe des parts aléatoires). Le produit scalaire sécurisé est utilisé pour l'addition des parts des distances, et le circuit d'évaluation de Yao est utilisé pour les opérations de comparaison sécurisée et de calcul des centroïdes intermédiaires. Ces primitives de sécurité sont exécutées pour chaque item de l'ensemble de données, ce qui renvoie le protocole au même problème du protocole proposé par J. Vaidya, celui d'un coût élevé de calcul pour un ensemble de données large. La solution présente une faiblesse lors de la mise à jour des centroïdes où l'on considère la division comme une multiplication par l'inverse, ce qui n'implémente pas correctement l'algorithme k-means. Un autre protocole a été donné par C. Su et al. [101] où ils introduisent un processus sécurisé de normalisation des données afin de donner plus d'exactitude au résultat du clustering. Une amélioration en sécurité par rapport au premier protocole est que la comparaison des distances et le calcul des moyennes, se fait par l'évaluation polynomiale aveuglée [98] et une technique d'approximation sécurisée [102]. Cependant la solution proposée ne résout pas le problème de la division locale, qui fait connaître le nombre d'entités dans chaque cluster aux différentes parties et la révélation des affectations intermédiaires des items aux clusters.

P. Bunn et R. Ostrovsky [103] proposent une solution de privacy assez complète pour l'algorithme k-means distribué et l'applique sur ce modèle de données. Les auteurs donnent une interprétation rigoureuse à la sécurité dans le modèle semi honnête, leur protocole ne révèle pas les centroïdes et les affectations des clusters intermédiaires et résout efficacement le problème de la division locale. Ils développent des sous protocoles pour chaque opération dans l'algorithme k-means en se basant sur les crypto systèmes homomorphes de Paillier [85] en particulier pour la division multi partie sécurisée. Malgré ça, leur solution faillit à rendre confidentiel le nombre d'itérations dans l'algorithme. Les crypto systèmes homomorphes ont aussi une complexité considérable.

Dans le même contexte, J. Sakuma et S. Kobayashi [104] donnent un nouveau protocole mais appliqué sur plusieurs parties, qu'ils appellent des nœuds. L'objectif est de donner un protocole de préservation de privacy entre utilisateurs au lieu d'organisations ou compagnies. Dans ce cas, le protocole est scalable par rapport au Kowalsczyk et al. [105] qui calcule la moyenne des valeurs distribuées à travers des réseaux P2P sans transférer les données dans un dépositaire central pour le calcul sécurisé des centroïdes. A ce niveau, la privacy est préservée en utilisant les crypto systèmes homomorphes à clé publique de Paillier [85]. Pour la protection des parts des distances, l'évaluation du circuit de Yao et le principe des parts aléatoires sont introduits. Le calcul se fait pour chaque nœud, ce qui rend le protocole proposé scalable et insensible aux pannes. Mais, le problème de la révélation d'extra information persiste toujours du moment où l'affectation des clusters intermédiaires est connue aux parties.

VI. Résumé de l'analyse des travaux de préservation de privacy dans l'algorithme k-means :