Préservation de la vie privée

Recherche de motifs séquentiels dans des bases de données distribuées

Vishal Kapoor* — **Pascal Poncelet**** — **François Trousset**** **Maguelonne Teisseire*****

Netaji Subhas Institute of Technology Azad Hind Fauj Marg Dwarka New Delhi-110045, India

Vishal.Kapoor@trilogy.com

** *EMA-LGI2P/Site EERIE Parc Scientifique Georges Besse, F-30035 Nîmes cedex*

{pascal.poncelet, francois.trousset}@ema.fr

*** *LIRMM UMR CNRS 5506, 161 Rue Ada, F-34392 Montpellier cedex 5*

teisseire@lirmm.fr

RÉSUMÉ. *La nécessité de garantir la confidentialité des données sources lors de l'extraction de connaissance est un domaine de recherche actuel de la communauté fouille de données. Dans cet article, nous présentons un nouvel algorithme,* PRIPSEP *(privacy preserving sequential patterns), pour extraire des motifs séquentiels dans des bases de données distribuées tout en assurant la contrainte de préservation de la vie privée. Nous prouvons que notre architecture et les protocoles employés par notre algorithme sont sécurisés.*

ABSTRACT. *Extracting knowledge without disclosing any individual or sensitive information is a new challenging problem for the data mining community. In this paper, we present a new algorithm* PRIPSEP *(privacy preserving sequential patterns) for the mining of sequential patterns from distributed databases while preserving privacy. We prove that our architecture and protocols employed by our algorithm are secure.*

MOTS-CLÉS : *fouille de données, motifs séquentiels, préservation de la vie privée.*

KEYWORDS: *data mining, sequential patterns, privacy preserving.*

DOI:10.3166/ISI.12.1.85-107

1. Introduction

Ces dernières années, l'utilisation croissante des systèmes multibases a entraîné le développement d'un grand nombre de bases de données transactionnelles distribuées. Dans un contexte d'aide à la décision, les grandes organisations souhaitent alors pouvoir extraire de la connaissance à partir de l'ensemble de ces bases. Par exemple, si nous considérons une chaîne de magasins avec différentes franchises, chacune des bases transactionnelles peut contenir des informations sur l'historique des achats d'un même ensemble de clients. Fouiller les données en considérant l'union de toutes les bases transactionnelles offre de nouvelles connaissances utiles pour le décideur. Toutefois, même si ces gros volumes de données doivent permettre d'améliorer la qualité de la décision, nous sommes confrontés à la difficulté d'identifier efficacement des connaissances à partir de ces sources de données multiples (Wu *et al.*, 2003; Zhong *et al.*, 1999). En effet, à l'heure actuelle, les algorithmes de fouille de données considèrent que les données sont toutes stockées sur un même site centralisé.

Récemment, de nouvelles lois, comme HIPAA (*Health Insurance Portability and Accountability Act*) (Services, 1996) qui instaure un régime de protection des renseignements personnels en matière de santé aux Etats-Unis (ces lois sont à l'heure actuelle adoptées par de nombreux pays : Australie, Chine, Japon...) ou les nouvelles directives européennes, imposent de nouvelles contraintes sur la confidentialité des données afin de préserver la vie privée des personnes. Bien entendu ce problème de confidentialité peut être adapté à de nombreux domaines d'applications (analyses de transactions financières, analyses de comportements sur des sites de e-commerce...). Préserver la vie privée dans un contexte de fouille de données nécessite de n'offrir des connaissances que si celles-ci garantissent de ne pas divulger d'information sensible sur les individus concernés. Pour garantir que les algorithmes de fouille de données ne violent pas la vie privée des individus, certaines approches ont considéré qu'elles disposaient d'une connaissance préalable sur ce qui était sensible ou non. Cependant ce type d'approches reste très subjectif et est très difficile à mettre en œuvre.

Dans cet article, nous nous intéressons à l'extraction de motifs séquentiels dans des bases de données distribuées tout en préservant la vie privée sans connaissance *a priori*. Non seulement les approches existantes ne prennent pas en compte la contrainte de confidentialité mais également elles ne sont pas adaptées à de multiples sources de données. Traditionnellement les protocoles de calcul distribué sécuritaires multiparties ont été utilisés pour calculer de manière sécurisée n'importe quelle fonction générique. Cependant, la complexité de tels protocoles fait qu'ils ne sont pas adaptés à des tâches de fouille de données comme l'extraction de séquences. Dans cet article, nous présentons un nouvel algorithme, PRIPSEP (*privacy preserving sequential patterns*), pour extraire des motifs séquentiels dans des bases de données distribuées tout en respectant la contrainte de préservation de la vie privée. Nous proposons également l'architecture sécurisée associée qui est constituée de sites semi-honnêtes, *i.e.* qui suivent le protocole correctement mais sont libres d'utiliser l'information qu'ils ont collectée pendant l'exécution du protocole et ils ne collaborent pas entre eux (Kantarcioglu *et al.*, 2002).

Le reste de l'article est organisé de la manière suivante. La section 2 présente en détail la problématique. Dans la section 3, nous proposons un survol des travaux liés et donnons nos motivations pour une nouvelle approche. PRIPSEP est décrit dans la section 4. Finalement, la section 5 conclut le papier et propose différentes perspectives.

2. Problématique

Dans cette section, nous définissons formellement la problématique de la préservation de la vie privée lors de la recherche de motifs séquentiels dans des bases de données distribuées. Dans un premier temps, nous proposons un rapide survol du problème de l'extraction de motifs séquentiels tel qu'il a été introduit dans (Agrawal *et al.*, 1995) et (Srikant *et al.*, 1996). Nous étendons ensuite le problème en considérant des bases de données distribuées et la contrainte de préservation de la vie privée.

2.1. *Recherche de motifs séquentiels*

Le problème de la recherche de séquences dans une base de données de transactions est présenté dans (Agrawal *et al.*, 1995) de la façon suivante.

Définition 1 Une *transaction* constitue, pour un client C, l'ensemble des items achetés par C à une même date. Dans une base de données client, une transaction s'écrit sous la forme d'un ensemble : {id-client, id-date, itemset}. Un *itemset* est un ensemble d'items non vide noté $(i_1 i_2 \dots i_k)$ où i_j, avec j de 1 à k, est un *item* (il s'agit de la représentation d'une transaction non datée). Une *séquence* est une liste ordonnée, non vide, d'itemsets notée $< s_1 s_2 \dots s_n >$ où s_j est un itemset (une séquence est donc une suite de transactions qui apporte une relation d'ordre entre les transactions). Une *séquence de données* est une séquence représentant les achats d'un client. Soit $T_1,\ T_2, \dots,\ T_n$ les transactions d'un client, ordonnées par dates d'achat croissantes et soit itemset(T_i) l'ensemble des items correspondants à T_i, alors la séquence de données de ce client est $< itemset(T_1)\ itemset(T_2) \dots\ itemset(T_n) >$.

Exemple 1 *Soient C un client et S=$<$ (3) (4 5) (8) $>$, la séquence de données représentant les achats de ce client. S peut être interprétée par « C a acheté l'item 3, puis en même temps les items 4 et 5 et enfin l'item 8 ».*

Définition 2 Soient $s_1 =< a_1 a_2 \dots a_n >$ et $s_2 =< b_1 b_2 \dots b_m >$ deux séquences de données, s_1 est *incluse* dans s_2 ($s_1 \prec s_2$) si et seulement si il existe $i_1 < i_2 < \dots < i_n$ des entiers tels que $a_1 \subseteq b_{i_1},\ a_2 \subseteq b_{i_2}, \dots\ a_n \subseteq b_{i_n}$.

Exemple 2 *La séquence $s1 =<$ (3) (4 5) (8) $>$ est incluse dans la séquence $s2 =<$ (7) (3 8) (9) (4 5 6)(8) $>$ (*i.e. *$s1 \prec s2$) car* (3) $\subseteq$ (3 8), (4 5) $\subseteq$ (4 5 6) *et* (8) $\subseteq$ (8). *En revanche* $<$ (3) (5) $>\not<$ (3 5) $>$ *(et* vice versa*).*

Définition 3 Un client *supporte* une séquence s (fait partie du support pour s) si s est incluse dans la séquence de données de ce client. Le *support* d'une séquence s est calculé comme étant le pourcentage des clients qui supportent s. Soit σ le support minimal fixé par l'utilisateur, une séquence qui vérifie le support minimal (*i.e.* dont le support est supérieur ou égal à σ) est une *séquence fréquente*.

Définition 4 Soient une base de données DB et un support minimal σ, la problématique de la recherche de motifs séquentiels consiste à trouver l'ensemble des séquences vérifiant la contrainte de support minimal σ dans DB.

2.2. *Bases de données distribuées et préservation de la vie privée*

Soit DB une base de données telle que $DB = DB_1 \bigcup DB_2 ... \bigcup DB_D$. Par simplicité, nous considérons que toutes les bases $DB_1, DB_2 ... DB_D$ partagent le même nombre de clients (CIDs), appelé N, et les mêmes identifiants. Nous considérons également que pour chaque client, le nombre de transactions, K, est le même[1]. Enfin, dans la suite de cet article, nous considérons que les séquences sont réduites à une liste d'items par souci de simplification.

Définition 5 Soient DB une base de données telle que $DB = DB_1 \bigcup DB_2 ... \bigcup DB_D$ et un support minimal σ, la problématique de la recherche de motifs séquentiels dans des bases de données distribuées consiste à trouver l'ensemble des séquences vérifiant la contrainte de support minimal σ dans l'union des bases $DB_1 \bigcup DB_2 ... \bigcup DB_D$.

Considérons à présent que les bases DB_1, DB_2, ... DB_D ne peuvent pas échanger leur contenu à l'extérieur.

Définition 6 Soient DB une base de données telle que $DB = DB_1 \bigcup DB_2 ... \bigcup DB_D$ et un support minimal σ, la problématique de préservation de la vie privée lors de la recherche de motifs séquentiels dans des bases de données distribuées consiste à trouver l'ensemble des séquences vérifiant la contrainte de support minimal σ dans l'union des bases $DB_1 \bigcup DB_2 ... \bigcup DB_D$ tout en garantissant qu'aucune base ne fournisse directement d'information sur son contenu.

De manière à illustrer ces différentes problématiques considérons l'exemple suivant.

Exemple 1 *Soient trois bases de données Alice, Bob et Carol partageant les mêmes clients mais sur des sites différents. A chaque item est associée une estampille temporelle (cf. tableau 1). Par exemple, en associant à chaque item sa base d'origine*

1. Ce nombre commun est utilisé uniquement pour simplifier l'écriture des algorithmes et cette contrainte peut facilement être relâchée.

(cf. tableau 2 où l'exposant indique l'initiale de la base d'origine de l'item), la séquence du client 1 *est la suivante :* $<(1)_1^A\ (2)_2^B\ (7)_4^C\ (3)_5^A>$.

CID	Alice	Bob	Carol
1	$(1)_1\ (3)_5$	$(2)_2$	$(7)_4$
2	$(2)_4$	$(1)_3$	$(3)_6$
3	$(2)_6\ (3)_7$		$(1)_2\ (7)_3$

Tableau 1. *Un exemple de bases de données distribuées triées par CID*

CID	Séquences
1	$(1)_1^A\ (2)_2^B\ (7)_4^C\ (3)_5^A$
2	$(1)_3^B\ (2)_4^A\ (3)_6^C$
3	$(1)_2^C\ (7)_3^C\ (2)_6^A\ (3)_7^A$

Tableau 2. *Séquences de chacun des clients dans l'union de toutes les bases*

Considérons que la valeur de support minimal soit fixée à 50 %. Nous remarquons que l'item (1) n'est pas fréquent dans chaque base prise de manière individuelle. Cependant, en considérant l'union de toutes les bases (cf. tableau 2), nous obtenons la séquence fréquente suivante : $< (1)(2)(3) >$. *En appliquant la contrainte de préservation de la vie privée, ni Alice, ni Bob, ni Carol ne peuvent divulguer d'informations sur leurs clients. Dans ce cas, la difficulté est de savoir si une séquence est fréquente ou non. Ainsi, dans notre cas, Alice ne peut pas dire qu'elle dispose du client* 1 *et que ce dernier a acheté l'item* (1) *au temps* 1.

3. Travaux antérieurs

Dans cette section, nous présentons les différents travaux de recherche liés aux motifs séquentiels et à la préservation de la vie privée.

3.1. *Les approches de recherche de motifs séquentiels*

De nombreuses approches efficaces ont été proposées pour extraire, à partir de grandes bases de données, les motifs séquentiels (e.g. (Srikant *et al.*, 1996; Masseglia *et al.*, 1998; Zaki, 2001; Pei *et al.*, 2001)).

Même s'il existe des approches utilisant des projections récursives (e.g. (Han *et al.*, 2000; Pei *et al.*, 2001)), la plupart des approches (e.g. (Masseglia *et al.*, 1998; Zaki, 2001)) utilisent une approche de type « Générer-Elaguer » inspirée de la méthode Apriori (Agrawal *et al.*, 1993) : création de candidats (*Générer*), suivi du test de ces

<(1 2)(3)> <(1 2) (3)>

<(2) (3 4)> <(2) (3) (5)>

<(1 2) (3 4)> <(1 2) (3) (5)>

Figure 1. *Opérations de jointure : I-extension (gauche) et S-extension (droite)*

		(1)
C_1	T_1	0
	T_2	0
	T_3	1
	T_4	0
	T_5	1

Figure 2. *Représentation par vecteur de bits dans* SPAM

candidats pour confirmer leur fréquence dans la base (*Elaguer*). Contrairement aux itemsets, les candidats sont générés soit par S-extension, *i.e.* ajout d'une séquence, soit par I-extension, *i.e.* ajout d'un itemset (cf. figure 1). Dans SPAM (Ayres *et al.*, 2002), la génération est similaire mais tire partie d'une représentation verticale par bitmap de la base de données à la fois pour la génération des candidats mais également pour le calcul du support. Ainsi la séquence d'un client $C_1 = < (1)_3\ (1)_5 >$ est transformée sous la forme d'un vecteur de bits (cf. figure 2). La *S-extension* transforme le bitmap à étendre en mettant 0 à la date de la première occurrence de l'item et 1 à tous les bits suivants. La seconde étape consiste à appliquer un AND logique entre les vecteurs de bits. La figure 3 illustre comment le candidat <(1) (2)> est généré. Le comptage des candidats se fait simplement en ajoutant 1 à chaque fois qu'il existe au moins un bit à 1 dans le vecteur final. La *I-extension* est simplement réalisée *via* un AND logique.

(1)
1
0
0
0

$S\text{-}Extension \Longrightarrow$

$(1)_s$
0
1
1
1

$\bigwedge$

2
1
1
1
0

résultat $\Longrightarrow$

(1)(2)
0
1
1
0

Figure 3. *S-extension dans* SPAM

Toutes les approches d'extraction de séquences considèrent que les données sont regroupées dans une seule base de données et ne sont donc pas adaptées à notre contexte.

3.2. *Calcul distribué sécuritaire multipartie*

La problématique du calcul distribué sécuritaire multipartie (SMC - *Secure Multi-Party Computation*) est de permettre le calcul d'une fonction quelconque sur un ensemble de données réparties entre plusieurs entités. Chaque entité possède une partie des données et le calcul doit être réalisé de manière à ce qu'aucune des parties ne puisse déduire de quelque manière que ce soit les données des autres entités à partir des résultats du calcul et de ses propres données. Les études dans ce domaine ont été initiées par (Yao, 1982; Yao, 1986) pour une approche bipartie. Elles ont ensuite été généralisées à un nombre quelconque de parties (e.g. (Du *et al.*, 2001; Goldreich, 2000; Chaum *et al.*, 1988)). Dans (Goldreich, 2000; Chaum *et al.*, 1988), les auteurs ont prouvé qu'un tel calcul sécurisé peut être effectué pour toute fonction polynômiale. Dans leur approche, ils utilisent le fait qu'à toute fonction polynômiale, un circuit binaire réalisant cette fonction peut être associé. Ainsi, il suffit de montrer qu'il est possible de réaliser un calcul sécurisé pour chacune des portes logiques pour prouver leur argumentation. Même si les auteurs ont montré que théoriquement cette approche pouvait être utilisée pour résoudre différentes tâches de fouilles de données, elle est cependant difficilement réalisable lorsque l'on manipule de gros volumes de données. En effet, le circuit généré est rapidement très complexe et sa taille dépend directement de celle de ses entrées (*i.e.* une simple multiplication nécessite un circuit de taille quadratique par rapport à la taille de ses entrées). Cette approche devient donc prohibitive en termes de temps et d'espace. L'autre difficulté de mise en œuvre est liée à la nécessité que toutes les parties soient présentes (connectées) pendant toute la durée du processus. Le détail de ces limitations est décrit dans (Kantarcioglu *et al.*, 2002).

3.3. *Préservation de la vie privée et fouille de données*

Récemment, de nombreux travaux se sont intéressés à la définition d'algorithmes de fouille de données préservant la vie privée (e.g. (Pinkas, 2002; Clifton *et al.*, 2003)). Ainsi, des approches adaptées à la classification (Du *et al.*, 2004), à la recherche de règles d'association (Evfimievski *et al.*, 2004) et au clustering (Jagannathan *et al.*, 2005) permettent d'extraire de la connaissance tout en respectant la contrainte de la vie privée. Par exemple, (Lindell *et al.*, 2002) proposent une nouvelle approche de classification en utilisant un protocole de transfert sécurisé, inspiré des SMC (*Secure Multi-Party Computation*), entre les différentes bases de données. (Vaidya *et al.*, 2002) considèrent une architecture sécurisée composée de deux parties effectuant les opérations avec les différentes bases afin de rechercher les règles d'association. Enfin, une nouvelle architecture garante de la sécurité et particulièrement adaptée aux problèmes de fouille de données a été proposée dans (Kantarcioglu *et al.*, 2002).

A notre connaissance, seuls les travaux de (Zhan *et al.*, 2004) ont abordé la problématique de la recherche de motifs séquentiels tout en préservant la vie privée. Dans leur approche, les données de chaque base sont transformées. Un protocole sécurisé est alors exécuté pour rechercher les séquences. Théoriquement cette approche est valide et robuste. Cependant, elle souffre de deux lacunes liées aux contraintes sur les données. Tout d'abord les items ne peuvent appartenir qu'à une seule base. Ainsi, en considérant notre exemple précédent et en ne prenant pas en compte la possibilité d'avoir des items partagés entre les différentes parties, nous n'obtenons pas le résultat final. Un item comme (1), qui n'est pas supporté par suffisamment de clients dans une base individuelle n'apparaîtra pas dans les résultats finaux. Cette contrainte est également une sérieuse limitation pour les applications réelles dans lesquelles le partage d'items est un impératif (e.g. entreprises sur plusieurs sites, e-commerce). Le second problème est lié à la représentation des données séquentielles : le même client achetant le même item plus d'une fois dans la même base mais avec un TID différent n'est pas autorisé. Cette dernière contrainte réduit considérablement le type de séquence recherché.

4. L'approche PriPSeP

Dans cette section, nous proposons notre nouvelle approche pour extraire des motifs séquentiels dans des bases de données distribuées tout en respectant la contrainte de préservation de la vie privée. Nous nous intéressons tout d'abord à la problématique de l'extraction dans des bases de données distribuées de manière à expliquer notre méthodologie. Cette approche est ensuite étendue à la prise en compte de la contrainte de préservation de la vie privée. Finalement nous proposons un nouvel algorithme avec les protocoles sous-jacents qui sont supportés par une architecture sécurisée.

4.1. *Recherche de motifs dans des bases de données distribuées*

4.1.1. *Présentation générale*

Toutes les transactions sont représentées sous la forme de bitmaps verticaux (vecteurs) comme dans l'approche SPAM (Ayres *et al.*, 2002).

Définition 7 *Soit V_i^j un vecteur où j et i correspondent respectivement au i^{ieme} item et à la j^{ieme} base de données. V_i^j est défini de la manière suivante : $V_i^j = [C_1^{i,j}...C_N^{i,j}]$ où pour $u \in \{1..N\}$, $C_u^{i,j} = [T_1^{i,j,u}...T_K^{i,j,u}]$. $T_{v=\{1..K\}}^{i,j,u}$ correspond à la liste des itemsets du client u, de la base DB_j, pour l'item i. Il s'agit d'un tableau de bits de longueur K dont le v^{ieme} bit vaut 1 si le client u a acheté l'item i dans la base DB_j.*

Comme nous l'avons vu dans l'exemple 1 de la section 2, la difficulté de l'extraction dans des données distribuées réside dans le fait que nous devons traiter différentes

bases dans lesquelles l'ordre des items n'est pas connu à l'avance (e.g. l'item (7) du client de CID 1 dans la base de Carol intervient avant l'item (3) de la base de données d'Alice).

Nous considérons par la suite que nous disposons d'un algorithme d'extraction de séquences « à la Apriori » qui alterne entre phase de génération et vérification. Nous considérons que la génération des candidats est réalisée de manière traditionnelle en combinant des k-1 séquences fréquentes pour générer des k-séquences candidates (e.g. la phase de génération de candidats de GSP (Srikant *et al.*, 1996)). Nous étendons la phase de vérification de la manière suivante. Comme nous travaillons avec des bases distribuées, nous considérons que l'algorithme d'extraction peut interroger les D bases de données d'origine de manière à obtenir un vecteur correspondant à l'item spécifique i, *i.e.* $V_i^{[1..D]}$ pour toutes les séquences candidates.

De manière à illustrer le fonctionnement de notre approche, considérons deux bases de données DB_1 et DB_2. Ces bases contiennent trois clients et chaque client possède cinq transactions (TID). Soit la séquence candidate $< (1)(2) >$, nous cherchons à connaître le support de cette séquence dans l'ensemble de tous les clients pour les deux bases de données. Tout d'abord, nous extrayons de DB_1 le vecteur correspondant à l'item (1), *i.e.* V_1^1, et de DB_2 le vecteur V_1^2 (cf. partie gauche de la figure 4). Deux opérations sont ensuite réalisées sur ces vecteurs : (i) la première opération consiste à regrouper ces deux vecteurs en un seul, et (ii) lors de la seconde opération nous transformons le résultat précédent de manière à vérifier s'il peut être suivi par l'item (2). Les deux vecteurs sont fusionnés en appliquant l'opérateur binaire OR : $(\vee) : V_1^1 \vee V_1^2$. Pour la seconde opération, de manière similaire au processus de S-extension de l'algorithme SPAM (cf. section 3), nous considérons une fonction qui transforme le vecteur (bitmap). Cependant, étant donné que nous travaillons avec différentes bases de données et pour des raisons d'efficacité, nous considérons que ces deux opérations sont réalisées *via* la fonction f définie ci-après pour obtenir un nouveau vecteur $Z_1 = f(V_1^1 \vee V_1^2)$.

Définition 8 *Considérons un vecteur V_i^j pour une base de données j et un item i tel que $V_i^j = [C_1^{i,j}...C_N^{i,j}]$ où pour $u \in \{1..N\}$, $C_u^{i,j} = [T_1^{i,j,u}, ..., T_K^{i,j,u}]$. K correspond au nombre de TID et N représente le nombre de CID. Par simplification, nous notons ce vecteur V. Soit $f : [0,1]^{N \times K} \rightarrow [0,1]^{N \times K}$ une fonction telle que : $f(V) = f(C_1...C_N) = [f_c(C_1)f_c(C_2)...f_c(C_N)]$. Pour chaque $u \in \{1..N\}$, nous avons : $f_c(C_u) = \left| \begin{array}{l} 0 \\ T_1^u \\ T_1^u \vee T_2^u \\ T_1^u \vee T_2^u \vee T_3^u \\ ... \\ T_1^u \vee ... \vee T_{k-1}^u \end{array} \right|$*
où $\vee$ est un opérateur binaire. Nous pouvons remarquer que $Length(V) = N \times K$, $Length(C_u) = K$ et $Length(f(V)) = N \times K$.
Soit $g : [0,1]^{N \times K} \rightarrow [0,1]^N$ une fonction telle que : $g(V) = g(C_1...C_N) = [g_c(C_1)g_c(C_2)...g_c(C_N)]$. Pour chaque $u \in \{1..N\}$, nous avons : $g_c(C_u) = 1$ s'il

existe au moins un bit ayant la valeur 1 dans les transactions du client. Nous pouvons noter que $Length(g(V)) = N$.

Les vecteurs correspondant à l'item (2) sont ensuite extraits de DB_1 et DB_2 (respectivement V_2^1 et V_2^2). De manière similaire à l'étape précédente, le vecteur ($Z_2 = V_2^1 \vee V_2^2$) est calculé. L'opérateur binaire $\wedge$ est alors utilisé pour calculer $Z_1 \wedge Z_2$. La fonction g permet de transformer les vecteurs résultats afin de pouvoir compter le support des séquences, *i.e.* $Z_3 = g(f(V_1^1 \vee V_1^2) \wedge (V_2^1 \vee V_2^2))$. Le vecteur résultant Z_3 a une longueur correspondant au nombre de clients, *i.e.* N. La dernière opération à réaliser est la somme du nombre de bits à 1 dans le vecteur Z_3. Ceci est réalisé *via* l'opération $\sum$.

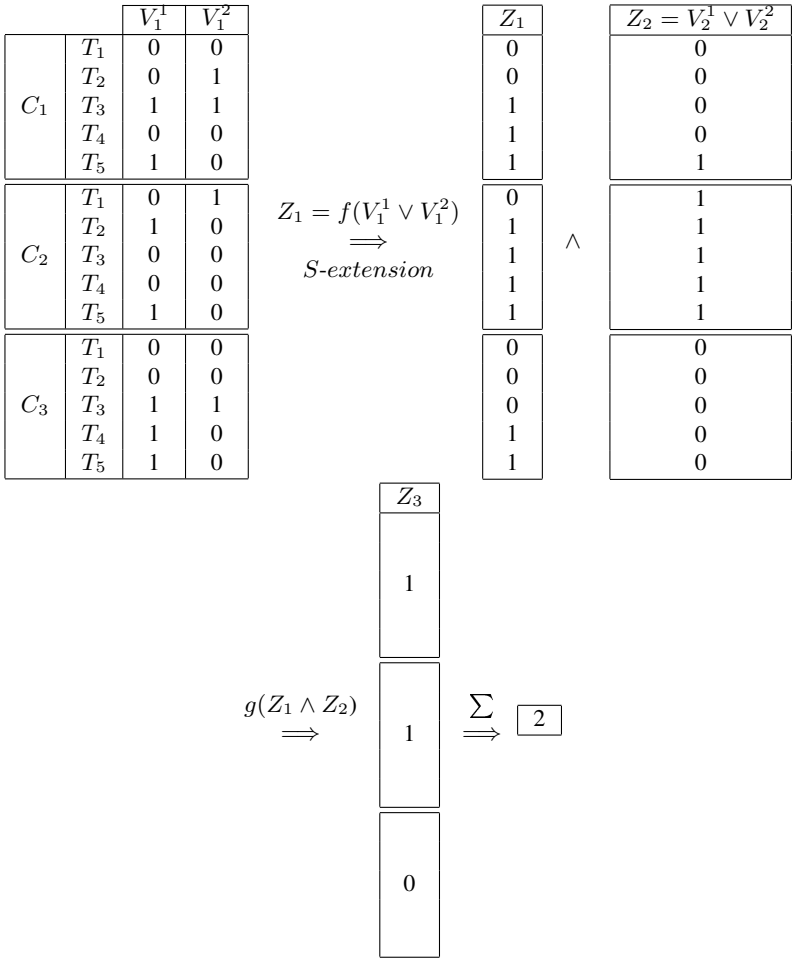

Figure 4. *Obtention des vecteurs dans le cas distribué*

4.1.2. *L'algorithme de comptage du support*

L'algorithme de calcul du support (cf. algorithme 1) fonctionne de la manière suivante. Pour chaque item i de la séquence candidate à tester, un nouveau vecteur X_i est généré en appliquant l'opérateur binaire sur tous les vecteurs des bases de données d'origine. En considérant le résultat de l'opération précédente, la fonction f est appliquée, suivie par l'opérateur binaire $\wedge$ pour chaque item. A la fin de cette itération, un nouveau vecteur Z de longueur $N \times K$ est produit. Puis, la fonction g est appliquée sur le résultat intermédiaire pour générer un vecteur de taille N, *i.e.* Y. Finalement, les bits à 1 dans Y sont sommés pour calculer la valeur finale du support.

Algorithme 1: L'algorithme de calcul du support pour des bases distribuées

Data: $S = < it_1 \ ... \ it_q >$ une séquence à tester ; $DB = DB_1 \bigcup DB_2 ... \bigcup DB_D$ un ensemble de bases de données ; N le nombre de clients partagés par toutes les bases ; K le nombre de dates (transactions) partagées par tous les clients de toutes les bases de données.

Result: Le support de la séquence S dans DB.

foreach $i \in 1..|S|$ **do**
  $X_i \leftarrow V^1_{it_i} \bigvee ... \bigvee V^D_{it_i}$;
$Z \leftarrow X_1$;
foreach $i \in 2..|S|$ **do**
  $Z \leftarrow f(Z) \bigwedge X_i$;
$Y \leftarrow g(Z)$;
return $\sum_{i=1}^{N} Y_i$;

Complexité. Soit $V_s = N \times K$ la taille des vecteurs qui sont envoyés et S la séquence candidate à vérifier. Les transferts qui sont réalisés par l'algorithme sont : $(V_s \times D \times |S|)$ pour $\bigvee$ et $(V_s \times |S|)$ pour la fonction f et l'opération $\bigwedge$. Il y a $(N(K-2))$ $\bigvee$ calculs réalisés par f. Si f est déjà disponible, *i.e.* calculé au préalable et stocké, nous avons (N) $\bigvee$ operations, autrement $(N(K-1))$ $\bigvee$ operations sont réalisées par g.

4.2. *Prise en compte de la contrainte de préservation de la vie privée*

4.2.1. *Un survol de l'architecture*

Dans cette section, nous décrivons une nouvelle architecture sécurisée associée à l'algorithme PRIPSEP, pour pouvoir répondre à la problématique de la préservation de la vie privée. Inspirée des travaux de (Kantarcioglu *et al.*, 2002) dans le domaine de la recherche d'itemsets, cette architecture offre l'avantage de pouvoir réaliser les différentes étapes de l'extraction de séquences tout en garantissant qu'aucune des parties ne

puisse avoir accès aux données privées des bases d'origine. Outre le site chargé d'effectuer les fonctions de fouille (Générer-Elaguer), l'architecture nécessite trois sites *non collaboratifs* et *semi-honnêtes* (Goldreich, 2000) : ils suivent le protocole correctement mais sont libres d'utiliser l'information qu'ils ont collectée pendant l'exécution du protocole. Ces sites indépendants collectent, stockent et évaluent l'information de manière sécurisée. Les différentes fonctions de ces sites sont les suivantes :

– *le site de fouille de données (Data Miner Site)* DM. Le site Data Miner est choisi aléatoirement parmi les bases de données d'origine. Son but est de réaliser les opérations de génération des candidats, d'interagir avec deux sites non collaboratifs NC_1 et NC_2, et de recevoir le résultat final du calcul envoyé par PS. Il faut noter que le résultat final est aussi envoyé à toutes les bases d'origine ;

– *les sites non collaboratifs (Non Colluding Sites)* NC_1 *et* NC_2. Ces sites symétriques collectent les données bruitées de toutes les bases (dont le site Data Miner) et réalisent une série d'opérations de manière sécurisée sans pouvoir inférer ni des résultats intermédiaires ni le résultat final ;

– *le site de calcul (Processing Site)* PS. Ce site est utilisé à la fois par NC_1 et NC_2 pour calculer de manière sécurisée les différentes fonctions et opérations. De manière similaire aux sites NC_1 et NC_2, PS ne peut pas déduire les résultats intermédiaires ou finaux des données qu'il traite.

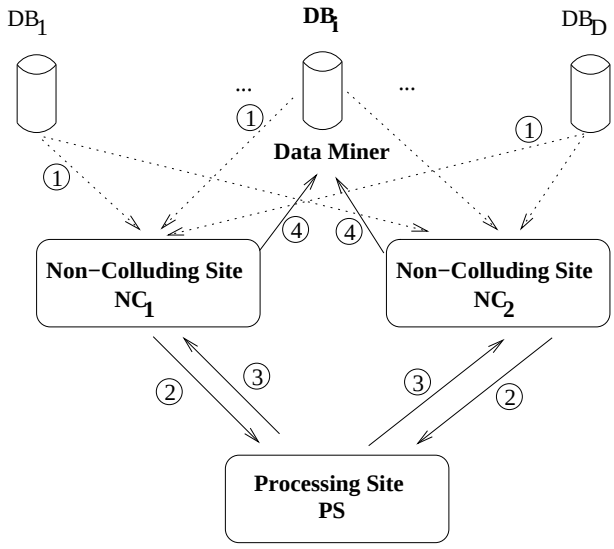

Figure 5. *L'architecture de* PRIPSEP

Les différents sites ainsi que les échanges effectués sont décrits dans la figure 5. Le processus débute en effectuant les étapes de prétraitement suivantes :

1) l'objectif de la première étape est de bruiter le nombre de clients des différentes bases de données afin qu'au cours du processus aucun des sites ne puisse connaître le nombre exact de client. Chaque base de données $DB_1, DB_2...DB_D$ ajoute un même nombre de clients ε. Les transactions des clients sont remplies de manière aléatoire. Un algorithme de comptage est alors utilisé sur les différentes bases pour connaître le nombre de clients ε' qui devront être supprimés du résultat final. Ce comptage peut être réalisé par n'importe quel algorithme car cette étape est indépendante de la contrainte de préservation de la vie privée ;

2) soit φ un nombre aléatoire. Chaque base permute individuellement ses vecteurs de transactions (V_i^j) en fonction de φ. L'objectif de cette étape est d'éviter que les différents sites puissent déduire l'ordre des différents clients ;

3) finalement, le site d'une base de données est choisi aléatoirement afin de jouer le rôle de site de Data Miner (DM). Son rôle est de récupérer les résultats, de générer les candidats et de lancer la vérification des candidats sur les autres bases de données.

A l'issue de ces étapes de prétraitement, nous disposons de bases de données ayant des faux clients et une liste permutée de vecteurs. Le site Data Miner peut alors appliquer un algorithme « à la Apriori » pour générer des séquences candidates comme cela a été décrit dans la section 4.1. Cette étape est directement suivie par l'étape de vérification des candidats sur les différentes bases. De manière à illustrer le principe général, considérons le calcul du support pour la séquence $< (1)(2) >$. Chaque base de données DB_j envoie son vecteur V_1^j à NC_1 et NC_2 (cf. flèche numérotée 1 dans la figure 5). De manière à minimiser les risques d'une attaque sur le réseau, nous proposons une fonction hypothétique $\text{SEND}^S \times DB_d(it)$ qui transmet de manière sécurisée le vecteur de l'item V_{it} de la base de données DB_d à NC_1 et NC_2. En outre, de manière à garantir que NC_1 et NC_2 reçoivent le minimum d'information, pour chaque base de données DB_i, nous générons un vecteur aléatoire R_{DB_i} ayant la même taille que V_{it} et nous calculons un nouveau vecteur : $Z_{DB_i} = V_{it} \bigoplus R_{DB_i}$. Z_{DB_i} est alors envoyé à NC_1 et R_{DB_i} à NC_2 (ou *vice versa*). Une approche similaire est proposée dans (Clifton *et al.*, 2003) pour d'autres types de tâches de fouille. Dans ce cas, les sites NC_1 et NC_2 ne disposent que de vecteurs aléatoires ou de vecteurs XOR-isés avec un vecteur aléatoire.

De manière similaire à l'algorithme de la section 4.1, l'opérateur binaire ($\vee$) est appliqué entre les différents vecteurs. Comme ces vecteurs sont partagés par NC_1 et NC_2, nous considérons un nouveau protocole $\bigvee^S$ (cf. flèche numérotée 2 dans la figure 5) qui réalise l'opérateur OR entre les différents vecteurs. L'objectif de ce protocole est d'envoyer des valeurs aléatoires XOR-isées de NC_1 et NC_2 à PS. Pour cela, NC_1 et NC_2 génèrent des nombres aléatoires qu'ils s'échangent et XOR-isent leur vecteur à l'aide de ces nombres avant de les transmettre. A son tour, PS ajoute du bruit dans les vecteurs résultants de manière à répartir le résultat final entre NC_1 et NC_2. Comme précédemment, le calcul se poursuit en appliquant les fonctions f et g (appelées f^S et g^S dans le cas sécurisé) et les résultats sont stockés également entre

NC_1 et NC_2 (cf. flèches numérotées 3 dans la figure 5). Finalement, de manière à calculer le nombre de bits qui sont à 1 (fonction $\sum$ maintenant appelée $\sum^S$), NC_1 et NC_2 collaborent à nouveau pour mettre à jour leur vecteur résultant avec des valeurs aléatoires et réordonnent le nouveau vecteur. PS peut alors calculer la somme du nombre de bits à 1 et retourne le résultat à NC_1 et NC_2. NC_1 peut alors supprimer le bruit aléatoire qu'il avait rajouté et retourner le résultat final au site Data Miner (cf. flèche numérotée 4 dans la figure 5). A ce niveau, DM doit seulement combiner les résultats venant de NC_1 et NC_2 et supprimer la valeur ε' correspondant au nombre de clients aléatoires ajoutés lors de la phase de prétraitement.

Dans les sections suivantes, nous décrivons en détail les différents protocoles, fonctions et algorithmes utilisés dans PRIPSEP. Tout d'abord nous introduisons différentes notations qui seront utilisés pour décrire les algorithmes. Comme nos fonctions utilisent des opérateurs binaires, nous présentons dans un premier temps les nouveaux protocoles pour réaliser de manière sécurisée les opérations binaires. Nous poursuivons notre présentation en montrant comment les fonctions f, g et $\sum$ sont étendues en fonction f^S, g^S et $\sum^S$ pour prendre en compte l'aspect sécurisé. Finalement, nous présentons l'algorithme général pour l'extraction de séquences dans des bases de données distribuées qui préserve la vie privée. Etant donné que notre objectif est de préserver la vie privée et de ne pas divulger d'information sur le résultat final aux différents sites de notre architecture, nous montrons qu'à la fin du processus NC_1, NC_2 et PS ne peuvent apprendre qu'une borne supérieure du support des séquences et qu'ils n'ont pas la possibilité d'obtenir d'information sur les données privées des différents clients.

4.2.2. *Notations*

De manière à simplifier l'écriture des algorithmes, nous considérons les notations suivantes. Soit $(\overset{+}{X} \mid \overset{-}{X}) \leftarrow h^S(\overset{+}{Y_1} \ldots \overset{+}{Y_n} \mid \overset{-}{Y_1} \ldots \overset{-}{Y_n})$ un calcul tripartie de n'importe quelle fonction h^S entre NC_1, NC_2 et PS où NC_1 possède une partie des entrées $\overset{+}{Y_1} \ldots \overset{+}{Y_n}$ et obtient une partie du résultat $\overset{+}{X}$, et de manière similaire NC_2 possède une partie des entrées $\overset{-}{Y_1} \ldots \overset{-}{Y_n}$ et obtient une partie des résultats $\overset{-}{X}$ à la fin du processus. Le résultat final est obtenu en appliquant l'opérateur binaire XOR ($\bigoplus$) entre $\overset{+}{X}$ et $\overset{-}{X}$. Cependant, cela n'implique pas que NC_1 envoie directement $\overset{+}{Y_1} \ldots \overset{+}{Y_n}$ à PS et reçoit le résultat $\overset{+}{X}$ de PS. En fait, NC_1 transforme ses entrées $\overset{+}{Y_1} \ldots \overset{+}{Y_n}$ en $\overset{+}{Y_1'} \ldots \overset{+}{Y_n'}$ *via* l'addition d'un bruit aléatoire uniforme et les envoie de manière sécurisée à PS. De manière symétrique, NC_2 envoie aussi ses entrées bruitées à PS. A la fin du calcul, les deux sites reçoivent le résultat partagé et bruité $\overset{+}{X'}$ et $\overset{-}{X'}$ de PS. Ce résultat intermédiaire pourra par la suite être utilisé comme entrée pour d'autres calculs.

Algorithme 2: Le protocole $\bigwedge^S$

Data: $(\overset{+}{X}, \overset{+}{Y} \mid \overset{-}{X}, \overset{-}{Y})$ sont des bits tels que $\overset{+}{X}$ et $\overset{+}{Y}$ appartiennent à NC_1, $\overset{-}{X}$ et $\overset{-}{Y}$ appartiennent à NC_2.

Result: $(A^R|B^R)$ sont tels que $A^R \bigoplus B^R = (\overset{+}{X} \bigoplus \overset{-}{X}) \bigwedge (\overset{+}{Y} \bigoplus \overset{-}{Y})$.

- NC_1 et NC_2 génèrent et échangent quatre nombres aléatoires R_A, R'_A, R_B et R'_B tels que : $\overset{+}{X}' = \overset{+}{X} \bigoplus R_A$, $\overset{+}{Y}' = \overset{+}{Y} \bigoplus R'_A$, $\overset{-}{X}' = \overset{-}{X} \bigoplus R_B$ et $\overset{-}{Y}' = \overset{-}{Y} \bigoplus R'_B$.
- NC_1 envoie $\overset{+}{X}'$ et $\overset{+}{Y}'$ à PS.
- NC_2 envoie $\overset{-}{X}'$ et $\overset{-}{Y}'$ à PS.
- PS calcule $\overset{+}{C} = \overset{+}{X}' \bigwedge \overset{-}{Y}'$ et $\overset{-}{C} = \overset{-}{X}' \bigwedge \overset{+}{Y}'$ ainsi qu'un nombre aléatoire R_{PS}.
- PS envoie $A'_{PS} = \overset{+}{C} \bigoplus R_{PS}$ à NC_1 et $B'_{PS} = \overset{-}{C} \bigoplus R_{PS}$ à NC_2.
- NC_1 calcule $A^R = A'_{PS} \bigoplus (\overset{+}{X} \bigwedge R'_B) \bigoplus (\overset{+}{Y} \bigwedge R_B) \bigoplus (\overset{+}{X} \bigwedge \overset{+}{Y}) \bigoplus (R_B \bigwedge R'_A)$
- NC_2 calcule $B^R = B'_{PS} \bigoplus (\overset{-}{X} \bigwedge R'_A) \bigoplus (\overset{-}{Y} \bigwedge R_A) \bigoplus (\overset{-}{X} \bigwedge \overset{-}{Y}) \bigoplus (R_A \bigwedge R'_B)$.

4.2.3. *Les protocoles $\bigwedge^S$ et $\bigvee^S$*

Dans cette section, nous définissons deux algorithmes ($\bigwedge^S$ (cf. algorithme 2) et $\bigvee^S$ (cf. algorithme 3)) décrivant le protocole utilisé pour réaliser de manière sécurisée les opérations binaires. Le principe fondamental de ces algorithmes est d'ajouter un bruit aléatoire uniforme aux données qui pourra par la suite être supprimé du résultat final. Le protocole débute avec NC_1 et NC_2 qui modifient leurs données en les XOR-isant avec des valeurs aléatoires. NC_1 et NC_2 échangent ces valeurs. Les données modifiées sont alors envoyées (e.g. pour NC_2, $\overset{-}{X}' = \overset{-}{X} \bigoplus R_B$ et $\overset{-}{Y}' = \overset{-}{Y} \bigoplus R'_B$) à PS, qui peut alors calculer de manière sécurisée l'opérateur $\bigwedge$ (resp. l'opérateur $\bigvee$). En fait, PS travaille sur des données bruitées et calcule $\overset{+}{C} = \overset{+}{X}' \bigwedge \overset{-}{Y}'$ et $\overset{-}{C} = \overset{-}{X}' \bigwedge \overset{+}{Y}'$. Ce dernier ajoute également un bruit aléatoire aux résultats intermédiaires afin d'éviter que NC_1 et NC_2 ne disposent du résultat final. A la fin du protocole, les sites non collaboratifs peuvent alors calculer le résultat final en supprimant le bruit qu'ils avaient ajouté. Par exemple, pour NC_1, l'opération suivante : $A^R = A'_{PS} \bigoplus (\overset{+}{X} \bigwedge R'_B) \bigoplus (\overset{+}{Y} \bigwedge R_B) \bigoplus (\overset{+}{X} \bigwedge \overset{+}{Y}) \bigoplus (R_B \bigwedge R'_A)$ peut être réalisée de manière sécurisée dans la mesure où il connaît ses propres éléments ($\overset{+}{X}$, $\overset{+}{Y}$ et R'_A) et les nombres aléatoires de NC_2 (R'_B et R_B). Le résultat final est alors $A^R \bigoplus B^R = A'_{PS} \bigoplus (\overset{+}{X} \bigwedge R'_B) \bigoplus (\overset{+}{Y} \bigwedge R_B) \bigoplus (\overset{+}{X} \bigwedge \overset{+}{Y}) \bigoplus (R_B \bigwedge R'_A) \bigoplus B'_{PS} \bigoplus (\overset{-}{X} \bigwedge R'_A) \bigoplus (\overset{-}{Y} \bigwedge R_A) \bigoplus (\overset{-}{X}$

$\bigwedge {}^{\bar{}}{}^{Y}) \bigoplus (R_A \bigwedge R'_B)$ où $A'_{PS} \bigoplus B'_{PS} = ({}^{+}_{X} \bigwedge R'_B) \bigoplus ({}^{+}_{Y} \bigwedge R_B) \bigoplus ({}^{\bar{}}{}^{X} \bigwedge R'_A) \bigoplus ({}^{\bar{}}{}^{Y} \bigwedge R_A) \bigoplus ({}^{+}_{X} \bigwedge {}^{+}_{Y}) \bigoplus ({}^{\bar{}}{}^{X} \bigwedge {}^{\bar{}}{}^{Y}) \bigoplus (R_A \bigwedge R'_B) \bigoplus (R_B \bigwedge R'_A) \bigoplus R_{PS} \bigoplus R_{PS}$.

Grâce à la propriété de l'opérateur XOR : $R \bigoplus R = 0$, nous obtenons le résultat désiré : $A^R \bigoplus B^R = {}^{+}_{X} \bigwedge {}^{+}_{Y} \bigoplus {}^{+}_{X} \bigwedge {}^{\bar{}}{}^{Y} \bigoplus {}^{\bar{}}{}^{X} \bigwedge {}^{+}_{Y} \bigoplus {}^{\bar{}}{}^{X} \bigwedge {}^{\bar{}}{}^{Y}$. Toutefois, cette opération n'est jamais réalisée par les sites non collaboratifs et le résultat final est partagé entre NC_1 et NC_2.

Algorithme 3: Le protocole $\bigvee^S$

Data: $({}^{+}_{X}, {}^{+}_{Y} \mid {}^{\bar{}}{}^{X}, {}^{\bar{}}{}^{Y})$ sont des bits tels que ${}^{+}_{X}$ et ${}^{+}_{Y}$ appartiennent à NC_1, ${}^{\bar{}}{}^{X}$ et ${}^{\bar{}}{}^{Y}$ appartiennent à NC_2.

Result: $(A^R|B^R)$ sont tels que $A^R \bigoplus B^R = ({}^{+}_{X} \bigoplus {}^{\bar{}}{}^{X}) \bigvee ({}^{+}_{Y} \bigoplus {}^{\bar{}}{}^{Y})$.

- Les 5 premières étapes sont similaires à celle de $\bigwedge^S$.
- NC_1 calcule

$$A^R = A'_{PS} \bigoplus ({}^{+}_{X} \bigwedge R'_B) \bigoplus ({}^{+}_{Y} \bigwedge R_B) \bigoplus {}^{+}_{X} \bigoplus {}^{+}_{Y} \bigoplus ({}^{+}_{X} \bigwedge {}^{+}_{Y}) \bigoplus (R_B \bigwedge R'_A).$$

- NC_2 calcule

$$B^R = B'_{PS} \bigoplus ({}^{\bar{}}{}^{X} \bigwedge R'_A) \bigoplus ({}^{\bar{}}{}^{Y} \bigwedge R_A) \bigoplus {}^{\bar{}}{}^{X} \bigoplus {}^{\bar{}}{}^{Y} \bigoplus ({}^{\bar{}}{}^{X} \bigwedge {}^{\bar{}}{}^{Y}) \bigoplus (R_A \bigwedge R'_B).$$

Propriété 1 *Le protocole $\bigwedge^S$ (resp. $\bigvee^S$) interdit à NC_1 d'apprendre des données privées de NC_2 et* vice versa*. En outre, la troisième partie PS ne peut rien apprendre sur leurs entrées privées.*

Preuve. A partir du protocole, B'_{PS} est tout ce que NC_2 peut apprendre sur les données privées de NC_1. Grâce au bruit ajouté R_{PS}, NC_2 ne peut pas trouver les valeurs de ${}^{+}_{X}$ ou ${}^{+}_{Y}$. Etant donné que les rôles de NC_1 et NC_2 sont interchangeables, la même argumentation peut s'appliquer à NC_1 qui ne peut pas apprendre les données privées ${}^{\bar{}}{}^{X}$ ou ${}^{\bar{}}{}^{Y}$ de NC_2. En outre, en bruitant leurs entrées, NC_1 et NC_2 garantissent qu'aucune information privée n'est transmise à PS. Enfin, grâce à l'étape de prétraitement, PS ne dispose que d'un flot de valeurs distribuées et il ne peut distinguer les vrais valeurs des valeurs aléatoires.

Complexité. Pour l'opérateur $\bigwedge^S$, dix opérations doivent être réalisées ($6 \bigoplus$ et $4 \bigwedge$). Etant donné que deux opérations XOR sont réalisées dans le protocole $\bigvee^S$, nous avons au total 12 opérations. Pour chaque $\bigwedge^S$, NC_1 et NC_2 échangent 2×2 bits. A partir de NC_1 ou NC_2, 2×1 bits sont envoyés à PS et un bit retourné. En outre, NC_1 et NC_2 calculent tous les deux 2 bits aléatoires et 1 bit aléatoire est généré par PS.

4.2.4. *Les fonctions f^S, g^S et $\sum^S$*

Algorithme 4: La fonction f^S

Data: Vecteurs de bits $(\overset{+}{X} \mid \overset{-}{X})$. $\overset{+}{X}$ vient de NC_1 et $\overset{-}{X}$ vient de NC_2. K le nombre de dates partagées par tous les clients de toutes les bases.

Result: Vecteurs $(\overset{+}{Y} \mid \overset{-}{Y})$ tels que $\overset{+}{Y}$ est la partie de NC_1 et $\overset{-}{Y}$ est la partie de NC_2. Le résultat final est obtenu en effectuant l'opération : $\overset{+}{Y} \bigoplus \overset{-}{Y}$.

foreach $c \in 0..(|\overset{+}{X}|/K) - 1$ **do**
  // Pour chaque client c dans [0..N-1]
  $(\overset{+}{Y}_{K\times c+1} \mid \overset{-}{Y}_{K\times c+1}) \leftarrow (0|0)$;
  foreach $i \in 2..K$ **do**
    $(\overset{+}{Y}_{K\times c+1} \mid \overset{-}{Y}_{K\times c+1}) \leftarrow$
    $\bigvee^S(\overset{+}{Y}_{K\times c+i-1}, \overset{+}{X}_{K\times c+i-1} \mid \overset{-}{Y}_{K\times c+i-1}, \overset{-}{X}_{K\times c+i-1})$;

return $(\overset{+}{Y} \mid \overset{-}{Y})$;

Algorithme 5: La fonction g^S

Data: Vecteurs de bits $(\overset{+}{X} \mid \overset{-}{X})$. $\overset{+}{X}$ vient de NC_1 et $\overset{-}{X}$ vient de NC_2. K le nombre de dates partagées par tous les clients de toutes les bases.

Result: Vecteurs $(\overset{+}{Y} \mid \overset{-}{Y})$ tels que $\overset{+}{Y}$ sera envoyé à NC_1 et $\overset{-}{Y}$ sera envoyé à NC_2. Le résultat final est obtenu en effectuant l'opération : $\overset{+}{Y} \bigoplus \overset{-}{Y}$.

foreach $c \in 0..(|\overset{+}{X}|/K) - 1$ **do**
  // Pour chaque client c dans [0..N-1]
  $(\overset{+}{Y_c} \mid \overset{-}{Y_c}) \leftarrow (\overset{+}{X}_{K\times c+1} \mid \overset{-}{X}_{K\times c+1})$;
  foreach $i \in 2..K$ **do**
    $(\overset{+}{Y_c} \mid \overset{-}{Y_c}) \leftarrow \bigvee^S(\overset{+}{Y_c}, \overset{+}{X}_{K\times c+i} \mid \overset{-}{Y_c}, \overset{-}{X}_{K\times c+i})$;

return $(\overset{+}{Y} \mid \overset{-}{Y})$;

Dans cette section, nous étendons les fonctions f et g dans un environnement sécurisé (cf. algorithme 4). Comme nous l'avons précisé précédemment, l'étape de S-extension de SPAM nécessite que les vecteurs correspondants à chaque client contiennent tous un 1 après la date de la première transaction du client. Ainsi, la fonction f^S emploie récursivement la fonction $\bigvee^S$ pour calculer en toute sécurité le vecteur résultat. Les entrées de la fonction sont les données clients bruitées et le protocole sécurisé $\bigvee^S$ est utilisé pour effectuer le OR binaire entre les bits successifs des deux sites NC_1 et NC_2. De manière similaire à l'algorithme précédent, le résultat final est découpé en deux parties par le site PS.

De manière similaire, la fonction g^S (cf. algorithme 5) calcule de manière sécurisée l'existance d'au moins un bit positionné à 1 dans le vecteur de chaque client. Il réduit le vecteur à un simple bit de valeur 0 ou 1. La valeur 1 est attribuée si la séquence est supportée au moins une fois. Cette fonction est utilisée lors de l'étape finale de calcul de la valeur du support dans l'algorithme 7.

Propriété 2 *Les fonctions f^S et g^S sont sécurisées et permettent d'éviter que NC_1, NC_2 et PS ne puissent inférer des informations sur des données privées.*

Preuve. A partir des différents algorithmes, il apparaît que l'opération sécurisée $\bigvee^S$ est appliquée de manière itérative pour arriver au résultat final. Comme il a été prouvé dans la Propriété 1, aucune information privée n'est partagée lors de l'exécution de cette opération. Ainsi, les fonctions f^S et g^S sont également sécurisées et aucun site ne peut extraire de l'information sur les différents clients.

Remarque. En fait, le calcul de $g^S(\overset{+}{X}, \overset{-}{X}) \rightarrow (\overset{+}{Y}, \overset{-}{Y})$ peut être retourné lorsque l'on calcule $f^S(\overset{+}{X}, \overset{-}{X}) \rightarrow (\overset{+}{Z}, \overset{-}{Z})$ car $\overset{+}{Y_i}, \overset{-}{Y_i}$ peut facilement être obtenu à partir de $(\overset{+}{Z_{i\times K+K}}, \overset{-}{Z_{i\times K+K}})$ en utilisant la relation suivante :
$(\overset{+}{Y_i} \mid \overset{-}{Y_i}) = \bigvee^S(\overset{+}{Z_{i\times K+K}}, \overset{+}{X_{i\times K+K}} \mid \overset{-}{Z_{i\times K+K}}, \overset{-}{X_{i\times K+K}})$.

Algorithme 6: Le protocole $\sum^S$

Data: Vecteurs de bits $(\overset{+}{X} \mid \overset{-}{X})$. $\overset{+}{X}$ qui vient de NC_1 et $\overset{-}{X}$ vient de NC_2.

Result: Un nombre qui est partagé en deux parties : $(\overset{+}{NB} \mid \overset{-}{NB})$, correspondant au nombre de bits à 1 dans les vecteurs $(\overset{+}{X} \bigoplus \overset{-}{X})$, tel que le résultat réel est $NB = \overset{+}{NB} + \overset{-}{NB}$.

• NC_1 et NC_2 génèrent et échangent deux vecteurs aléatoires R_1 et R_2 de même longueur tels que $(Length(R_1) = Length(R_2) \geq 2N)$. Ils calculent tous les deux $R_1 \bigoplus R_2$ et calculent le nombre de 1 qui doit être supprimé , N_R, à la fin du calcul de PS.

• NC_1 et NC_2 réordonnent respectivement les vecteurs ($\overset{+}{X}$, R_1) et ($\overset{-}{X}$, R_2) en utilisant une valeur de permutation φ et obtiennent respectivement $\overset{+}{Y}$ et $\overset{-}{Y}$.

• NC_1 envoie $\overset{+}{Y}$ à PS et NC_2 envoie $\overset{-}{Y}$ à PS.

• PS calcule $\overset{+}{Y} \bigoplus \overset{-}{Y}$ et compte le nombre de bits à 1 et obtient NB.

• PS génère un nombre aléatoire R_{PS} et retourne $\overset{+}{N} = NB + R_{PS}$ à NC_1 et $\overset{-}{N} = NB - R_{PS}$ à NC_2.

• NC_1 calcule $\overset{+}{NB} = \overset{+}{N} - N_R$, NC_2 conserve simplement $\overset{-}{NB} = \overset{-}{N}$.

Propriété 3 *Les fonctions du protocole $\sum^S$ sont totalement sécurisées et ne révèlent pas à NC_1, NC_2 ou PS la valeur finale du support pour une séquence candidate.*

Preuve. Deux vecteurs aléatoires R_1 et R_2 sont ajoutés aux entrées de NC_1 et NC_2 pour éviter que PS ne sache s'il travaille sur des données réelles ou aléatoires. Le résultat final calculé est divisé par PS entre les deux sites NC_1 et NC_2. Donc le calcul final est réalisé par Data Miner qui reçoit le résultat correct.

Complexité. Dans l'algorithme 6, le nombre de bits est augmenté d'une valeur $\geq 2N$ pour des raisons de sécurité. Considérons que nous positionnons cette valeur de la manière suivante $t =\in [2..K]$. Pour NC_1 et NC_2, $(2N(2t+1))$ opérations sont réalisées et $(2N(t+1))$ opérations sont effectuées sur PS. En outre, nous avons $N(t+1)$ opérations pour bruiter les données. Le nombre de transferts entre NC_1 et NC_2 est $(2tN)$. Pour échanger la permutation φ entre NC_1 et NC_2, nous avons en fait besoin de $N(t+1)$ transferts. Cependant, si NC_1 et NC_2 partagent un ensemble commun de générateurs de valeurs aléatoires, ils ont uniquement à s'échanger le nombre et la graine. Ce qui est négligeable. Finalement entre NC_1/NC_2 et PS, $N(t+1)$ bits sont transférés.

4.2.5. *L'algorithme de calcul du support sécurisé*

Algorithme 7: L'algorithme de calcul du support sécurisé

Data: $S =< it_1...it_q >$ une séquence candidate à tester ; $DB = DB_1 \bigcup DB_2... \bigcup DB_D$ un ensemble de bases de données ; N le nombre de clients partagés par toutes les bases de données ; K le nombre de dates partagées par tous les clients de toutes les bases.

Result: Le support de la séquence S dans DB avec un bruit aléatoire.

foreach $i \in 1..|S|$ **do**
  $(\overset{+}{X_i} \mid \overset{-}{X_i}) \leftarrow \text{SEND}^S \times DB_1(i)$;
  foreach $j \in 2..D$ **do**
    $(\overset{+}{V} \mid \overset{-}{V}) \leftarrow \text{SEND}^S \times DB_j(it_i)$;
    $(\overset{+}{X_i} \mid \overset{-}{X_i}) \leftarrow \bigvee^S (\overset{+}{C_i}, \overset{+}{V} \mid \overset{-}{C_i}, \overset{-}{V})$;

$(\overset{+}{Z} \mid \overset{-}{Z}) \leftarrow (\overset{+}{X_1} \mid \overset{-}{X_1})$;
foreach $i \in 2..|S|$ **do**
  $(\overset{+}{T} \mid \overset{-}{T}) \leftarrow f^S(\overset{+}{Z} \mid \overset{-}{Z})$;
  $(\overset{+}{Z} \mid \overset{-}{Z}) \leftarrow \bigwedge^S (\overset{+}{T}, \overset{+}{X_i} \mid \overset{-}{T}, \overset{-}{X_i})$;
$(\overset{+}{Y} \mid \overset{-}{Y}) \leftarrow g^S(\overset{+}{Z} \mid \overset{-}{Z})$;
$(\overset{+}{R} \mid \overset{-}{R}) \leftarrow \sum^S (\overset{+}{Y} \mid \overset{-}{Y})$;
return $(\overset{+}{R} \mid \overset{-}{R})$;

L'algorithme de calcul du support sécurisé (cf. algorithme 7) étend l'algorithme 1 de manière à réaliser toutes les opérations de manière sécurisée. Il est appliqué après l'étape de prétraitement et considère donc que les bases de données d'origine possèdent de fausses transactions. Pour chaque item i de la séquence à tester, tous les vecteurs bruités sont envoyés par SEND^S à NC_1 et NC_2 de manière à appliquer de manière sécurisée un OR entre chaque vecteur ($\bigvee^S$). La fonction f^S suivie par l'opérateur binaire $\bigwedge^S$ est réalisée. A la fin de la boucle, nous disposons d'un nouveau vecteur $(\overset{+}{Z} \mid \overset{-}{Z})$ où les parties des résultats sont partagées entre NC_1 et NC_2. Nous appliquons alors la fonction g^S pour générer $(\overset{+}{Y} \mid \overset{-}{Y})$. Finalement, nous comptons le nombre de bits à 1 dans $(\overset{+}{Y} \mid \overset{-}{Y})$ *via* la fonction $\sum^S$. A la fin du processus, $\overset{+}{R}$ et $\overset{-}{R}$ sont envoyés respectivement par NC_1 et NC_2 au site Data Miner. Pour obtenir le résultat réel et final, ce dernier doit simplement calculer $\overset{+}{R} + \overset{-}{R}$ (somme d'entiers) et supprimer le bruit aléatoire ajouté lors du prétraitement, *i.e.* ε'.

Propriété 4 *Le fait de rajouter du bruit à chaque étape (bases de données d'origine, NC_1, NC_2 et PS) n'affecte pas l'exactitude du résultat.*

Preuve. Les premiers bruits sont ajoutés dans les bases d'origine en insérant des faux clients, *i.e.* ε, et en permutant la liste des clients *via* la valeur de φ. Comme DM est choisi parmi les bases de données d'origine, il connaît les bruits ajoutés et peut donc facilement les supprimer. NC_1 et NC_2 ajoutent également du bruit lorsqu'ils envoient les vecteurs de transaction à PS pour le calcul sécurisé de $\bigvee^S$, $\bigwedge^S$, f^S et g^S. Ce bruit ajouté est supprimé à la fin de chaque calcul par NC_1 et NC_2 lorsqu'ils reçoivent les résultats de PS en réalisant un XOR avec les données aléatoires initiales. Nous avons prouvé en outre qu'aucune information sur les données individuelles ne peut être apprise par ces différents sites (cf. Propriétés 1, 2, et 3). Finalement, pour le calcul de la somme, *via* la fonction $\sum^S$, NC_1 et NC_2 ajoutent du bruit aléatoire sur leur donnée, *i.e.* N_R, et permutent également leur vecteur en fonction de la valeur φ. PS bruite également sa valeur entière et ce bruit est supprimé en envoyant les parties opposées à NC_1 et NC_2. La valeur N_R est supprimée par NC_1 et NC_2 lorsqu'ils retournent le résultat à DM. Finalement, lorsque les résultats de NC_1 et NC_2 sont combinés, la seule opération à réaliser par DM pour connaître le support exact est la suppression du bruit initial ε'.

Complexité. Dans le protocole sécurisé, chaque base de données envoie $2NK$ bits au lieu de NK. Chaque base doit aussi calculer NK bits aléatoires et réaliser (NK) $\bigoplus$ opérations. D'après les résultats précédents sur le nombre d'opérations réalisées par les opérateurs sécurisés, la complexité en temps est $O(12NK)$ pour les opérations binaires et $O(7NK)$ pour les opérations de création de nombres aléatoires. Donc, ceci est borné par $O(20NK)$. Considérons à présent la complexité de communication du protocole. Soit $p = D \times |S| \times N \times K$. La complexité de l'algorithme 1, *i.e.* sans prise en compte de la préservation de la vie privée, est linéaire. Considérons $C_{or} = O(p)$. Comme les algorithmes sécurisés utilisent les mêmes structures ainsi que le même

ordre des opérations, nous avons une complexité de $20 \times C_{or}$. Le nombre de transferts requis est donc au moins quatre fois la complexité du transfert de 1.

L'architecture sécurisée peut bien sûr être améliorée de manière à optimiser les coûts de communication entre NC_1, NC_2 et PS. En outre, toutes les fonctions peuvent facilement être parallélisées.

4.2.6. *Sécurité du protocole*

Pour analyser la sécurité, examinons les informations propagées à chaque site participant au protocole. Notons que pendant tout le processus, les nombres aléatoires sont générés de manière sécurisée et que l'infrastructure de communication est robuste et résistante aux intrusions.

– *Point de vue NC_1 et NC_2*. Lors de l'exécution du protocole, les deux sites voient simplement un flot de données aléatoires avec une distribution uniforme. *Via* le protocole proposé, ils reçoivent uniquement des données bruitées ou des résultats bruités partagés. Etant donné que, par définition, NC_1 et NC_2 sont des sites semi-honnêtes, ils ne peuvent pas partager d'information. Les valeurs reçues des différentes bases de données sont XOR-isées avec des nombres aléatoires ou correspondent à des nombres aléatoires.

– *Point de vue de PS*. Il réalise les différentes opérations sécurisées ($\bigwedge^S$, $\bigvee^S$, f^S, g^S, $\sum^S$) et transmet les résultats à NC_1 et NC_2. Comme nous l'avons vu précédemment (cf. Propriété 1, Propriété 2 et Propriété 3) toutes ces opérations et fonctions ne fournissent pas d'information sur les données privées des clients des bases de données d'origine. Même l'application répétée des protocoles reste sécurisée.

– *Sécurité générale*. Lors de l'application de l'algorithme dans son intégralité, aucun site ne peut apprendre plus que ce pourquoi il a été défini. Ainsi, la sécurité et la préservation de la vie privée de chaque client est maintenue lors du calcul du support dans l'architecture proposée. L'ajout de clients lors de l'étape de prétraitement et la permutation de la liste de clients permettent en outre de garantir que tous les sites ne peuvent obtenir ni résultat intermédiaire ni résultat final.

5. Conclusion

Dans cet article, nous avons abordé la problématique de l'extraction de motifs séquentiels dans des bases de données distribuées en respectant la contrainte de préservation de la vie privée. Nous avons présenté une nouvelle extension sécurisée de l'algorithme SPAM pour extraire des motifs. Nous avons également prouvé que, sous certaines conditions raisonnables (tiers semi-honnêtes), notre algorithme, ainsi que les opérations, protocoles et l'architecture sont sécurisés. Il existe de nombreux travaux possibles liés à cette proposition. Tout d'abord nous nous sommes focalisés sur la S-extension de l'algorithme SPAM, *i.e.* nous avons uniquement considéré le problème de la découverte de séquences réduites à une liste d'items. Comme nous avons proposé une série de fonctions sécurisées, notre approche peut facilement être éten-

due pour considérer également la I-extension, *i.e.* une liste d'itemsets plutôt qu'une liste d'items. De la même manière, les algorithmes et les protocoles associés peuvent facilement être étendus en considérant que le nombre de TID est différent entre les différents clients. Dans ce cas, la seule contrainte à considérer reste le fait que les bases de données doivent partager le même nombre de clients ou CID. En outre, dans la version courante de PRIPSEP, les résultats sont directement envoyés à DM ainsi qu'aux bases de données d'origine. De manière à améliorer le processus dans sa globalité, nous prévoyons d'étendre le rôle de DM en stockant un arbre lexicographique pour lequel chaque nœud de l'arbre pourra être étendu en considérant que les résultats intermédiaires sont stockés dans des tableaux temporaires entre NC_1 et NC_2. Ainsi, une fouille incrémentale peut être envisagée et les résultats intermédiaires n'ont plus besoin d'être recalculés. Le stockage de ces résultats peut bien entendu être sécurisé en garantissant que chaque site ne dispose que de données bruitées ou aléatoires.

6. Bibliographie

Agrawal R., Imielinski T., Swami A., « Mining association rules between sets of items in large database », *Proc. of ACM SIGMOD 93*, p. 207-216, 1993.

Agrawal R., Srikant R., « Mining sequential patterns », *Proc. of ICDE 95*, p. 3-14, 1995.

Ayres J., Flannick J., Gehrke J., Yiu T., « Sequential pattern mining using bitmap representation », *Proc. of KDD 02*, p. 439-435, 2002.

Chaum D., Crepeau C., Damgard I., « Multiparty unconditionally secure protocols », *Proc. of the 20th Annual Symposium on the Theory of Computing (STOC)*, p. 11-19, 1988.

Clifton C., Kantarcioglu M., Lin X., Vaidya J., Zhu M., « Tools for privacy preserving distributed data mining », *SIGKDD Explorations*, vol. 4, n° 2, p. 28-34, 2003.

Du W., Atallah M. J., « Secure multi-party computation problems and their applications : a review and open problems », *New Security Paradigms Workshop*, Cloudcroft, USA, p. 11-20, 2001.

Du W., Han Y., Chen S., « Privacy-preserving multivariate statistical analysis : linear regression and classification », *Proc. of the Fourth SIAM Int. Conf. on Data Mining*, p. 222-233, 2004.

Evfimievski A., Srikant R., Agrawal R., Gehrke J., « Privacy-preserving mining of association rules », *Information Systems*, vol. 29, n° 4, p. 343-364, June, 2004.

Goldreich O., « Secure multi-party computation - working draft », *citeseer.ist.psu.edu/goldreich98secure.html*, 2000.

Han J., Pei J., Mortazavi-asl B., Chen Q., Dayal U., Hsu M., « FreeSpan : frequent pattern-projected sequential pattern mining », *Proc. of KDD 00*, p. 355-359, 2000.

Jagannathan G., Wright R., « Privacy-preserving distributed k-means clustering over arbitrarily partitioned data », *Proc. of KDD 05*, p. 393-399, august, 2005.

Kantarcioglu M., Vaidya J., « An architecture for privacy-preserving mining of client information », *Proc. of the Workshop on Privacy, Security, and Data Mining in conjunction with the 2002 IEEE ICDM Conf*, p. 27-42, 2002.

Lindell Y., Pinkas B., « Privacy preserving data mining », *Journal of Cryptology*, vol. 15, n° 2, p. 177-206, 2002.

Masseglia F., Cathala F., Poncelet P., « The PSP approach for mining sequential patterns », *Proc. of PKDD 98*, p. 176-184, 1998.

Pei J., Han J., Pinto H., Chen Q., Dayal U., Hsu M., « PrefixSpan : mining sequential patterns efficiently by prefix projected pattern growth », *Proc. of ICDE 01*, p. 215-224, 2001.

Pinkas B., « Cryptographic techniques for privacy preserving data mining », *SIGKDD Explorations*, vol. 4, n° 2, p. 12-19, 2002.

Services H. . H., « United States Dept. of Health & Human Services : Health Insurance Portability and accountability of 1996 », *http ://www.hipaa.org/*, August, 1996.

Srikant R., Agrawal R., « Mining sequential patterns : generalizations and performance improvements », *Proc. of EDBT 96*, p. 3-17, 1996.

Vaidya J., Clifton C., « Privacy preserving association rule mining in vertically partitioned data », *Proc. of KDD 02*, p. 639-644, July, 2002.

Wu X., Zhang S., « Synthesizing high-Frequency rules from different data sources », *IEEE Trans. on Knowledge and Data Engineering*, vol. 15, n° 2, p. 353-367, 2003.

Yao A. C., « Protocols for secure computations », *Proc. of the 23rd annual IEEE Symposium on Foundations of Computer Science*, p. 160-164, 1982.

Yao A. C., « How to generate and exchange secrets », *Proc. of the 27th Symposium on Foundations of Computer Science (FOCS)*, p. 162-167, 1986.

Zaki M., « SPADE : An efficient algorithm for mining frequent sequences », *Machine Learning Journal*, vol. 42, n° 1, p. 31-60, February, 2001.

Zhan J., Chang L., Matwin S., « Privacy-Preserving Collaborative Sequential Pattern Mining », *Workshop on Link Analysis, Counter-terrorism, and Privacy in conjunction with SIAM Int. Conf. on Data Mining*, Lake Buena Vista, Florida, p. 61-72, 2004.

Zhong N., Yao Y., , Ohsuga S., « Peculiarity oriented multi-database mining », *Proc. of PKDD 99*, p. 136-146, 1999.