Recherche approximative de plus proches voisins efficace et sûre

Benjamin MATHON[1], Teddy FURON[1], Laurent AMSALEG[2], Julien BRINGER[3]

[1]INRIA [2]IRISA - CNRS
Campus de Beaulieu, 35042 Rennes, France

[3]MORPHO - SAFRAN Group
11 boulevard Gallieni, 92130 Issy-les-Moulineaux, France

benjamin.mathon@inria.fr, teddy.furon@inria.fr
laurent.amsaleg@irisa.fr, julien.bringer@morpho.com

Résumé – Cette communication présente une méthode de recherche approximative des plus proches voisins (APPV) modérément sûre mais très efficace. Nous partons d'une méthode de recherche APPV se basant sur des distances entre données quantifiées calculées à l'avance : les PQ-codes. Se plaçant dans le modèle « honnête mais curieux » dans lequel le serveur et l'utilisateur suivent le protocole mais sont capables de déduire de l'information sur les données transitées, nous analysons les menaces en terme de fuite d'information pour cette méthode. Pour empêcher une reconstruction du signal requête par le serveur, nous proposons une version des PQ-codes utilisant deux quantificateurs distincts pour le serveur et l'utilisateur. Les avantages de notre méthode sont (a) une perte significative d'information de la requête par le serveur (b) une base de taille fixée (c) aucune perte dans la qualité de la recherche.

Abstract – This paper presents a moderately secure but very efficient approximate nearest neighbors search. Our approach starts from a state-of-the-art algorithm in the domain of approximate nearest neighbors search: the PQ-codes. After detailing the threats belonging to the "honest but curious" model, where the server and the user follow the protocol, but are able to infer information about the transited data, we analyze the threats from information leakage for this method. To prevent a reconstruction of the query vector by the server, we propose a version of the PQ-codes using two different quantizers for the server and the user. The advantages of our method are (a) a significant loss of information of the query by the server (b) the database at the server side is fixed (c) any loss in the quality of research.

1 Introduction

La recherche des k plus proches voisins (k-PPV) consiste à trouver les k éléments les plus proches (au sens d'une distance donnée, bien souvent euclidienne) d'un vecteur requête dans une base. Dans ce domaine d'étude, le défi a longtemps été la réduction de la complexité : l'objectif étant de trouver rapidement les plus proches voisins d'une requête dans une base de n éléments, n étant très grand ($10^6 - 10^9$). De nombreux travaux de recherche se basent sur une recherche approximative des plus proches voisins (k-APPV) permettant de calculer rapidement les distances entre une requête et tous les éléments de la base : c'est le cas des PQ-codes. Les PQ-codes [3] (*Product Quantization codes*) permettent une recherche k-APPV efficace : la requête et les éléments de la base sont découpés en tronçons et chaque tronçon est ensuite quantifié sur un dictionnaire de K centres. La distance finale est finalement approchée grâce à la table des distances entre centres, pré-calculée à l'avance.

Récemment, d'autres défis ont été soulevés dans ce domaine : la sécurité et le respect de la vie privée [6]. Le vecteur requête appartient à l'utilisateur, la base appartient au propriétaire et aucun d'entre eux n'est prêt à dévoiler ses données. Ce cas se produit par exemple en biométrie. Le principal axiome en biométrie est qu'aucune base ne peut être stockée en toute sécurité. Par conséquent, un serveur ne peut avoir de base de modèles biométriques en clair étant donné que ces données pourraient être volées par un adversaire. De la même manière, un utilisateur est réticent à l'envoi de son modèle biométrique en clair.

La recherche des plus proches voisins est également le pivot de certains algorithmes de classification. Une classe est associée à chaque vecteur de la base et le but est de prédire la classe du vecteur requête à partir de la classe de ses voisins les plus proches. Cependant, un propriétaire peut être réticent à partager sa base de vecteurs et de classes, ces données étant issues d'un travail de collecte et de classification de sa part. L'utilisateur est intéressé par la valeur de prédiction, mais ne veut pas divulguer son vecteur requête pour des questions de confidentialité. C'est le cas des applications de diagnostics médicaux (les vecteurs sont ici issus de données médicales comme des électrocardiogrammes) ou des applications de recherche basée sur le contenu. Un utilisateur est ici intéressé par les contenus multimédia (images, vidéos, sons) similaires à sa requête tout en voulant s'assurer de ne pas trop divulguer d'informations sur sa personne, informations pouvant être liées à la nature de cette requête. Cette technologie est aussi utilisée dans les systèmes de gestion des droits numériques (*DRM*) permettant d'empê-

Ces travaux ont été supportés par le projet ANR-12-CORD-0014 SecuLar.

cher automatiquement l'envoi de contenus soumis aux droits d'auteur sur des plateformes multimédia. Les ayants droit des contenus ne souhaitent pas divulguer les caractéristiques extraites de ceux-ci tandis que l'utilisateur ne souhaite pas que les ayants droit sachent qu'il possède un contenu piraté.

Il existe déjà des solutions assurant la sécurité de la recherche des plus proches voisins basée sur le chiffrement : les méthodes par chiffrement homomorphique [4], l'insertion de Hamming [1] ainsi que le chiffrement basé sur les caractéristiques [5]. De notre point de vue, ces solutions mettent la sécurité et la confidentialité au dessus de la liste des exigences en sacrifiant l'efficacité et la vitesse de la recherche. Ces critères peuvent en effet être majeurs dans certaines des applications citées plus haut. L'utilisation d'éléments trop fortement sûrs peut être inutile, voire nuisible si ils dégradent d'autres caractéristiques du système, comme l'efficacité et la vitesse de la recherche.

Cet article présente une recherche approximative des plus proches voisins plutôt sûre mais hautement efficace et rapide. Nous partons de la méthode des PQ-codes, nous analysons les menaces côté serveur et côté utilisateur en termes de sécurité et de confidentialité pour cette méthode puis nous en proposons une version améliorée permettant de se prémunir au mieux contre ces menaces tout en évitant de pénaliser l'efficacité et la rapidité de la version classique. La section expérimentale utilise une base de taille beaucoup plus grande que ce que les dernières solutions sûres peuvent manipuler.

2 Les PQ-codes

Nous considérons un propriétaire possédant une collection de n vecteurs dans $\mathbb{R}^d$: $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$. Ces vecteurs sont alors scindés en M tronçons de longueur ℓ. Nous adoptons $d = M\ell$ et notons $\mathbf{x}_i^{(m)} = (\mathbf{x}_i((m-1)\ell+1), \ldots, \mathbf{x}_i(m\ell))$ le m-ième sous-vecteur de $\mathbf{x}_i$. Pour tout $m \in [M] = [\![1, M]\!]$, le propriétaire exécute un *K-means* sur les sous-vecteurs de $\mathcal{X}^{(m)} = \{\mathbf{x}_i^{(m)}\}_{i\in[n]}$. Ce procédé consiste à générer de façon aléatoire K vecteurs dans $\mathbb{R}^\ell$ et d'appliquer l'algorithme de Lloyd-Max jusqu'à convergence. Nous obtenons en sortie un dictionnaire de K centres $\mathcal{C}^{(m)} = \{\mathbf{c}_i^{(m)}\}_{i\in[K]}$ permettant de définir le m-ième quantificateur $Q^{(m)}(.) : \mathbb{R}^\ell \to [K]$:

$$Q^{(m)}(\mathbf{x}^{(m)}) = \arg\min_{i\in[K]} \|\mathbf{x}^{(m)} - \mathbf{c}_i^{(m)}\|, \quad \forall \mathbf{x}^{(m)} \in \mathbb{R}^\ell, \quad (1)$$

où $\|\cdot\|$ désigne la distance euclidienne. Le *K-means* converge vers un minimum local de la distorsion induite par l'erreur de reconstruction $\sum_{\mathbf{x}\in\mathcal{X}^{(m)}} \|\mathbf{x} - Q^{(m)}(\mathbf{x})\|^2$. Le propriétaire applique alors ce procédé sur un ensemble d'entrainement, sous-espace de $\mathcal{X}^{(m)}$. Nous définissons le quantificateur global $Q(\cdot) : \mathbb{R}^d \to [K]^M$ comme le quantificateur produit :

$$Q(\mathbf{x}) = (Q^{(1)}(\mathbf{x}^{(1)}), \ldots, Q^{(M)}(\mathbf{x}^{(M)})), \quad \forall \mathbf{x} \in \mathbb{R}^d, \quad (2)$$

et $Q^{-1}(\cdot) : [K]^M \to \mathbb{R}^d$, l'opérateur qui affecte à une séquence d'indices la concaténation de centres :

$$Q^{-1}((k_1, \ldots, k_M)) = \left(\mathbf{c}_{k_1}^{(1)\top} \ldots \mathbf{c}_{k_M}^{(M)\top}\right)^\top. \quad (3)$$

La base de données $\mathcal{Q} = \{Q(\mathbf{x}_i)\}_{i\in[n]}$ et l'ensemble des M dictionnaires $\mathcal{C} = \{\mathcal{C}^{(m)}\}_{m\in[M]}$ est alors envoyé au serveur. Le rôle du propriétaire est terminé. Le serveur pré-calcule les distances au carré entre centres du même dictionnaire :

$$d_s(i,j,m) = \|\mathbf{c}_i^{(m)} - \mathbf{c}_j^{(m)}\|^2, \forall (i,j,m) \in [K]\times[K]\times[M]. \quad (4)$$

La matrice d_s sera utilisée comme table de recherche. Lorsque le serveur reçoit une requête $\mathbf{q}$ d'un utilisateur, il calcule en premier lieu le quantifié $Q(\mathbf{q})$. La recherche APPV est ensuite basée sur la distance au carré approchée :

$$\hat{D}(\mathbf{q}, \mathbf{x}_i) = \|Q^{-1}(Q(\mathbf{q})) - Q^{-1}(Q(\mathbf{x}_i))\|^2, \quad (5)$$

à la place de la vraie distance $\|\mathbf{q} - \mathbf{x}_i\|^2$. Ce calcul est effectué rapidement grâce à la table de recherche :

$$\hat{D}(\mathbf{q}, \mathbf{x}_i) = \sum_{m=1}^{M} d_s(Q^{(m)}(\mathbf{q}^m), Q^{(m)}(\mathbf{x}_i^m), m). \quad (6)$$

Le serveur envoie à l'utilisateur les indices $(i_1, \ldots, i_k)$ correspondant aux k plus petites distances approchées.

3 Analyse des menaces

Nos travaux adoptent le modèle « honnête mais curieux » dans lequel le serveur et l'utilisateur suivent le protocole mais sont capables de déduire de l'information sur les données qu'ils reçoivent. Plus précisément, le serveur curieux peut vouloir :

- S_1 Reconstruire $\mathbf{x}_i$ à partir de $Q(\mathbf{x}_i)$,
- S_2 Regrouper les vecteurs de la base à partir de $Q(\mathbf{x}_i)$ (par recherches APPV parmi ces vecteurs),
- S_3 Reconstruire la requête $\mathbf{q}$ à partir des informations données par l'utilisateur,
- S_4 Détecter les requêtes similaires.

L'utilisateur curieux peut vouloir :

- U_1 Savoir à l'avance si deux requêtes similaires $\mathbf{q}$ et $\mathbf{q}'$ renvoient le même sous-ensemble k-ANN,
- U_2 Explorer efficacement un large voisinage de $\mathbf{q}$ en soumettant des requêtes quasi-similaires (par exemple en modifiant uniquement un tronçon).

Nous insistons sur le fait que ce travail ne propose pas de méthode qui se prémunit contre toutes ces menaces. Nous présentons dans la section suivante une méthode modérément sûre qui fait le compromis entre l'efficacité de la recherche APPV et la protection contre ces menaces.

4 Notre méthode

L'idée principale est ici d'éviter les menaces précédentes par l'introduction de deux quantificateurs. Le propriétaire génère hors ligne $\mathcal{C}_S$, un ensemble de M dictionnaires de K_S centres chacun. Cela définit le quantificateur produit $Q_S(\cdot)$ utilisé pour

générer la base $\mathcal{Q} = \{Q_S(\mathbf{x}_i)\}_{i=1}^n$ qui sera ensuite envoyée au serveur. Seul le propriétaire connait $\mathcal{C}_S$.

Le propriétaire génère également $\mathcal{C}_U$, un ensemble de M dictionnaires de K_U centres chacun, définissant le quantificateur $Q_U(\cdot)$. $\mathcal{C}_U$ sera ensuite envoyé à l'utilisateur pour quantifier sa requête $\mathbf{q}$. Le propriétaire calcule les distances :

$$d_{us}(i,j,m) = \|\mathbf{c}_{U,i}^{(m)} - \mathbf{c}_{S,j}^{(m)}\|^2,\ \forall (i,j,m) \in [K_U]\times[K_S]\times[M], \quad (7)$$

et envoie cette table de recherche au serveur.

En ligne, l'utilisateur récupère $\mathcal{C}_U$, envoie $Q_U(\mathbf{q})$ au serveur qui effectue la recherche APPV grâce à d_{us}. Notons que les deux quantificateurs peuvent ne pas avoir le même nombre de centres par sous-espace. Il est important d'avoir un K_S raisonnable en raison de l'empreinte mémoire de $\mathcal{Q}$ qui est de $nM \log_2 K_S$ bits.

Le serveur ne peut pas reconstruire les vecteurs de la base (menace S_1) car il ne connait pas le dictionnaire $\mathcal{C}_S$. Idem pour les vecteurs requêtes (menace S_3) car il ne possède pas $\mathcal{C}_U$. Notons que ces affirmations restent correctes tant qu'il n'existe pas de coalition entre le serveur et l'utilisateur et tant que le serveur n'usurpe pas le rôle de l'utilisateur (ces deux cas sortent du modèle « honnête mais curieux »). Le serveur peut détecter les requêtes similaires $\mathbf{q}$ et $\mathbf{q}'$ ($Q_U(\mathbf{q}) \approx Q_U(\mathbf{q}')$, menace S_4). Cependant, il peut difficilement jauger cette différence car il ne possède pas la table des distances entre les centres de $\mathcal{C}_U$.

5 Expériences

Nos tests sont effectués sur la base de descripteurs SIFT locaux *ANN_SIFT1M* [3]. Cette base contient (a) une base de $1,000,000$ vecteurs de dimension $d = 128$, (b) $100,000$ vecteurs d'entrainement pour générer les K-*means*, (c) $10,000$ vecteurs requête et un fichier de confiance qui contient, pour chaque requête, les identifiants de ses plus proches voisins triés par ordre croissant des distances.

5.1 Qualité de la recherche

Les PQ-codes réalisent une recherche APPV, ce qui signifie que les plus proches voisins retournés ne sont pas forcément les bons. Pour mesurer la qualité de cette recherche, le premier rappel au rang R, noté « 1-rappel@R » [3] est calculé. Cette quantité représente la probabilité pour que le premier vrai plus proche voisin soit contenu dans les R vecteurs retournés. La figure 1 montre les 1-rappel@R en pourcentages. Côté serveur, les PQ-codes sont calculés avec $M = 16,\ l = 8,\ K_S = 256$. La courbe en tirets montre les performances des PQ-codes classiques. En bref, la recherche retourne le plus proche voisin de manière quasi-certaine pour $R = 100$. Nous augmentons le nombre de centres pour le quantificateur utilisateur (de $K_U = 64$ à 4096) pour une meilleur qualité de recherche lorsque $K_U > K_S$. En général, le nombre de centres côté serveur est une puissance de deux afin que l'empreinte mémoire de la base soit de $nM \log_2 K_S$ bits, ce qui représente une

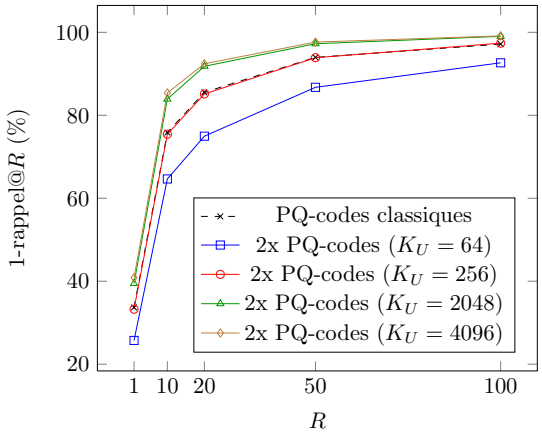

FIGURE 1 – Scores 1-rappel@R pour la version originale et proposée des PQ-codes : $M = 16,\ l = 8,\ K_S = 256$.

version très compacte de $\mathcal{X}$. Le temps de réponse est linéaire en nM. Pour nos paramètres $n = 10^6$, $M = 16$, $K_S = 256$, la base $\mathcal{Q}$ occupe 16Mo. Une fois que $Q_U(\mathbf{q})$ est calculé, une recherche APPV est exécutée sous 30ms (microprocesseur Core i7, une seule unité d'exécution). Le paramètre K_U n'a la plupart du temps aucun impact sur le temps de réponse, sous réserve que d_{us} soit stocké en mémoire.

5.2 Analyse de la menace S_2

Le serveur a besoin de connaître la table de recherche d_s pour calculer les distances entre les entrées de $\mathcal{Q}$. Il peut tout de même obtenir de l'information sur les plus proches voisins des vecteurs de la base en créant la table de recherche suivante :

$$d_p(i,j,m) = 1 - \delta_{i,j},\ \forall (i,j,m) \in [K]\times[K]\times[M], \quad (8)$$

où $\delta_{i,j}$ est le symbole de Kronecker ($= 1$ si $i = j$, 0 sinon). Le serveur peut toutefois obtenir une meilleure précision lors de la recherche grâce à la table d_{us} : si $d_{us}(i,j,m)$ est proche de 0 alors $\mathbf{c}_{U,i}^{(m)}$ est proche de $\mathbf{c}_{S,j}^{(m)}$. La distance $d_{us}(i,k,m)$ peut alors être prise comme une bonne estimation de $d_s(j,k,m)$. La table de recherche estimée $\hat{d}_s$ est alors construite de la façon suivante :

$$\begin{aligned} \hat{d}_s(j,k,m) &= (d_{us}(I(j),k,m) + d_{us}(I(k),j,m))/2, \\ \text{avec } I(j) &\triangleq \arg\min_{i\in[K_U]} d_{us}(i,j,m). \end{aligned} \quad (9)$$

La figure 2 montre les scores 1-rappel@R lorsque le serveur utilise (a) la table d_{us} (Eq. (7)), (b) la table d_p (Eq. (8)) et (c) la table estimée $\hat{d}_s$ (Eq. (9)) pour différentes valeurs de K_U. L'utilisation de d_p fournit une approximation grossière des k-APPV alors que les performances de la recherche en utilisant la table $\hat{d}_s$ sont légèrement inférieures que celles obtenues avec d_{us} (Fig. 2). Cela signifie que la menace S_2 ne peut être évitée. Notre méthode se rapproche alors de la recherche privée à sens unique [2] où l'accent est mis sur la confidentialité des données de l'utilisateur.

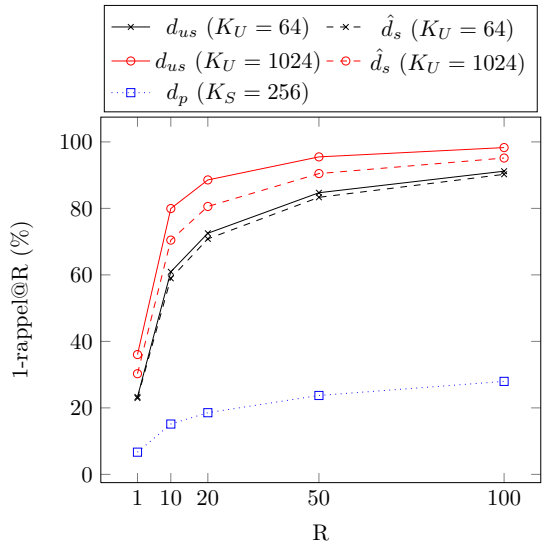

FIGURE 2 – Scores 1-rappel@R calculés pour différentes tables de recherche $\{d_{us},\ d_p,\ \hat{d}_s\}$ pour $M = 16$ et $K_S = 256$.

5.3 Analyse des menaces côté utilisateur

Lorsque l'utilisateur a accès au dictionnaire $\mathcal{C}_U$ et aux cellules de Voronoi associées à chaque sous-quantificateur, les menaces U_1 et U_2 peuvent être mises à exécution. Pour les éviter, un protocole sûr de calcul de distances est effectué : le serveur génère une paire de clés privée et publique (sk_S, pk_S) pour un chiffrement homomorphique $e(\cdot)$ (Paillier). Le propriétaire chiffre $e(\mathbf{c}_{U,i}^{(m)}, pk_S)$ et $e(\|\mathbf{c}_{U,i}^{(m)}\|^2, pk_S)$, pour tout $(i,m) \in [K] \times [M]$. Ces données sont alors envoyées à l'utilisateur qui calcule et envoie au serveur $e(D_i, pk_S)$ avec $D_i = \|\mathbf{q}^{(m)} - \mathbf{c}_{U,i}^{(m)}\|^2$. Le serveur déchiffre ces données et peut alors calculer $Q_U(\mathbf{q})$ sans avoir eu besoin de connaître $\mathbf{q}$ ni $\mathcal{C}_U$. De plus, l'utilisateur n'a pas connaissance de $Q_U(\mathbf{q})$. L'utilisateur et le serveur doivent alors calculer $M.K_U$ distances dans le domaine chiffré. Contrairement aux méthodes par chiffrement homomorphique citées en introduction, ce nombre est ici bien inférieur à n.

Cependant, le serveur est capable de connaître les distances intermédiaires D_i, ce qui rend la menace S_4 à nouveau possible. Afin d'éviter cette menace, l'utilisateur génère ici une paire de clés (sk_U, pk_U) pour un chiffrement homomorphique multiplicatif $E[\cdot]$ (El-Gamal).

Après avoir calculé $\forall m \in [M],\ e(D_i, pk_S)$, l'utilisateur envoie au serveur :

$$E[e(D_1, pk_S), pk_U], \ldots, E[e(D_{K_U}, pk_S), pk_U]. \quad (10)$$

Le serveur permute ces données afin de modifier leur ordre initial et calcule, grâce à l'homomorphisme multiplicatif de $E[\cdot]$:

$$E[e(D_{i_1}, pk_S)^\alpha, pk_U], \ldots, E[e(D_{i_{K_U}}, pk_S)^\alpha, pk_U], \quad (11)$$

avec $\alpha > 0$ tiré aléatoirement. Ce qui donne, grâce à l'homomorphisme additif de $e[\cdot]$:

$$E[e(\alpha.D_{i_1}, pk_S), pk_U], \ldots, E[e(\alpha.D_{i_{K_U}}, pk_S), pk_U]. \quad (12)$$

Le serveur envoie alors ces données à l'utilisateur qui les déchiffre sans connaître l'ordre original (le rôle de α est ici de masquer les données afin que l'utilisateur ne puisse deviner la permutation choisie par le serveur).

Finalement, l'utilisateur et le serveur exécutent un algorithme de tri interactif en comparant les distances dans le domaine chiffré en suivant le principe du problème des millionnaires de Yao. La comparaison sûre de deux données chiffrées $e(x, pk_S)$ et $e(y, pk_S)$ est effectuée grâce au procédé suivant. Soit R et R' deux nombres aléatoires tels que $R >> R'$. L'utilisateur calcule $e(R(x-y) - R', pk_S)$ grâce à l'homomorphisme. Le serveur déchiffre ce message et obtiendra $x > y$ si $R(x-y) - R' > 0$. Le serveur peut alors déterminer l'index de la distance minimale en effectuant $K_U - 1$ comparaisons successives avec l'utilisateur. L'ordre original étant uniquement connu du serveur, ce dernier obtient $Q_U^{(m)}(\mathbf{q}^{(m)})$. Ce calcul sûr a pour avantage de donner le minimum d'information au serveur et à l'utilisateur.

Nous évaluons maintenant la taille de la bande passante pour $K_U = 256$. Le propriétaire envoie les centres chiffrés ainsi que leurs normes à l'utilisateur, soit $M(\ell+1)K_U \times 2048$ bits (10Mo). L'utilisateur envoie au serveur les distances chiffrées par El Gamal, soit $MK_U \times 4096$ bits (2Mo). Ces distances sont ensuite renvoyées dans le désordre par le serveur, soit 2Mo à nouveau. Pour le protocole de Yao, l'utilisateur envoie $MK_U \times 2048$ bits (1Mo) au serveur. En terme de temps de calcul, l'utilisateur effectue $O(MK_U(\ell+3))$ exponentiations ($\sim$ 50s) et le serveur $O(2MK_U)$ ($\sim$ 8s).

Références

[1] P. BOUFOUNOS et S. RANE : Secure binary embeddings for privacy preserving nearest neighbors. *In Information Forensics and Security (WIFS), IEEE International Workshop on*, p. 1–6, 2011.

[2] G. FANTI, M. FINIASZ et K. RAMCHANDRAN : One-Way Private Media Search on Public Databases : The Role of Signal Processing. *IEEE Signal Processing Magazine*, 30(2):53–61, 2013.

[3] H. JÉGOU, M. DOUZE et C. SCHMID : Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):117–128, 2011.

[4] R. LAGENDIJK, Z. ERKIN et M. BARNI : Encrypted signal processing for privacy protection : Conveying the utility of homomorphic encryption and multiparty computation. *IEEE Signal Processing Magazine*, 30(1):82–105, 2013.

[5] S. RANE et W. SUN : An attribute-based framework for privacy preserving image querying. *In Image Processing (ICIP), 19th IEEE International Conference on*, p. 2649–2652, 2012.

[6] L. SANKAR, S. R. RAJAGOPALAN et H. V. POOR : Utility-Privacy Tradeoff in Databases : An Information-theoretic Approach. *IEEE Transactions on Information Forensics and Security*, 8(6), 2013.