Secure and Efficient Approximate Nearest Neighbors Search

Benjamin Mathon, Teddy Furon, Laurent Amsaleg, Julien Bringer

Secure and Efficient Approximate Nearest Neighbors Search

Benjamin Mathon
INRIA Rennes, France
benjamin.mathon@inria.fr

Teddy Furon
INRIA Rennes, France
teddy.furon@inria.fr

Laurent Amsaleg
IRISA - CNRS,
Rennes, France
laurent.amsaleg@irisa.fr

Julien Bringer
MORPHO - SAFRAN
Issy-les-Moulineaux, France
julien.bringer@morpho.com

ABSTRACT

This paper presents a moderately secure but very efficient approximate nearest neighbors search. After detailing the threats pertaining to the ‘honest but curious’ model, our approach starts from a state-of-the-art algorithm in the domain of approximate nearest neighbors search. We gradually develop mechanisms partially blocking the attacks threatening the original algorithm. The loss of performances compared to the original algorithm is mainly an overhead of a constant computation time and communication payload which are independent of the size of the database.

Categories and Subject Descriptors

H.2.0 [**Database Management**]: General—*Security, integrity, and protection*; H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models*

Keywords

Approximate Nearest Neighbors search; Privacy; Security.

1. INTRODUCTION

This paper deals with nearest neighbors search, an algorithm that finds the closest elements from a query vector within a database, according to a given distance metric. The main challenge in this field had been for a long time scalability: to retrieve the k nearest neighbors (in short k-NN) among a large database of n elements, n being extremely large ($10^6 - 10^9$), with a short time response. This challenge has been addressed in many research works proposing *approximate* nearest neighbors (k-ANN) search. The best solutions return some vectors which are likely to be the true nearest neighbors, striking a trade-off between efficiency and quality of search. There are mainly two ways. First, an approximate distance, which is faster to compute, is used instead of the given metric. Second, the database is indexed offline, i.e. it is partitioned into groups. The k-ANN is processed within the group the query vector belongs to. This speeds up the search because the cardinality of this group is smaller than n. Both solutions can be used independently or in conjunction. This article focuses on the first idea as it is based on the state-of-the-art k-ANN algorithm Product-Quantization codes (PQ-codes) [6].

Recently, other challenges have raised in this field: security and privacy. The query vector belongs to the User, the database to the Owner, and none of them is willing to share their property. This case happens for instance in biometrics identification. The main axiomatic in biometric claims that no database can be stored securely. Therefore, a Server cannot have the database of biometric templates in the clear since a pirate would steal these highly valuable data. In the same way, the User is reluctant in sending his biometric template in the clear.

The nearest neighbor search is also the pivot of some classification algorithms. A class is associated to each vector of the database, and the goal is to predict the class of the query vector from the class of its nearest neighbors. The Owner does not want to share its collection of pairs of vector and class, as this is the fruit of his know-how in collecting and assessing the quality of these data. The User is interested in the prediction value but does not want to disclose his query vector for some privacy issues. This happens in applications such as medical diagnostic (vectors are medical records like ECG) or user recommendation system (vectors are the user profiles). Another application is Content Based Retrieval where the User looks for multimedia contents (images, videos, audio clips) perceptually similar in some sense. This technology is now deeply used in Digital Right Management systems where copyright holders are reluctant in disclosing neither their contents nor the features extracted from their contents.

There are already solutions providing secure nearest neighbors search based on cryptographic primitives such as homomorphic encryption, oblivious transfer, argument based encryption, secure multiparty computation protocol. We provide a critical overview in Sect. 3.1. In brief, we believe that these solutions put security and privacy on top of the requirements list, sacrificing a lot the scalability and the speed of the search. Scalability and speed are of utmost importance in some applications and these past solutions are

IH&MMSec'13, June 17–19, 2013, Montpellier, France.

just not adequate here because they are too slow. Another point is that the security levels of these cryptographic primitives are very high, whereas, in some applications, they do not prevent some basic attacks on the global system. There is no use in rising big walls if the door is weakly secured. One motto in security is ‘A system is as secure as its weakest link’. This implies that using too strongly secure bricks is useless or even harmful if they degrade other features of the system, like scalability and speed in k-NN search.

This article presents a moderately secure but highly scalable and fast approximate nearest neighbors search. Our philosophy is to start from a state-of-the-art technique in this field, i.e. PQ-codes [6] presented in Sect. 3.2, to analyze the threats, and to patch it avoiding as much as possible bricks too much penalizing the scalability and the speed. On the other hand, we do not completely prevent the players to infer some knowledge, but these limitations are well explained and experimentally assessed. The experimental body uses database of size much bigger than what the past secure solutions can handle.

2. THE STARTING POINT

2.1 The framework

The framework considers an Owner having a collection of n pairs of a vector $\mathbf{x}_i \in \mathbb{R}^d$ and metadata t_i (defined in some space). Define $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$. The Owner subcontracts the k-NN (or k-ANN) search to an entity called the Server. For this purpose, the Owner gives a representation of each vector $h(\mathbf{x}_i)$ together with the metadata t_i (or an encrypted version of the metadata). The User has a query vector $\mathbf{q} \in \mathbb{R}^d$ and he is interested in some information about the subset $\mathcal{N}(\mathbf{q})$ of the k-NN of $\mathbf{q}$. Depending on the application, this can be their indices $(\mathcal{N}(\mathbf{q}))$, the values of these vectors $(\{\mathbf{x}_i\}_{i\in\mathcal{N}(\mathbf{q})})$, or their metadata $(\{t_i\}_{i\in\mathcal{N}(\mathbf{q})})$.

For instance, in a classification application, the metadata t_i is the class associated to the vector and the prediction of the class of $\mathbf{q}$ is a function of the classes of the k-ANN vectors. In a Content Based retrieval scenario, t_i is the ID of the content from which the feature vector $\mathbf{x}_i$ has been extracted. By a voting mechanism, the most similar contents' ID are detected. In a biometric identification problem, the metadata is the user ID. An exhaustive k'-NN search over the returned k-ANN $(k' < k)$ can also refine the result. The paper does not deal with this extension.

2.2 The threats

Our work adopts the ‘honest but curious’ model where the Server and the User follow the protocol but they might be willing to infer more information from what they know. More precisely, we explicitly list the potential threats under this model. The curious Server might want to:

S_1 Reconstruct $\mathbf{x}_i$ from $h(\mathbf{x}_i)$,

S_2 Cluster the database vectors from $\{h(\mathbf{x}_i)\}$ (i.e. by running k-ANN among vectors of the database),

S_3 Reconstruct $\mathbf{q}$ from what it receives from the User,

S_4 Detect similar queries (from one or different Users).

The curious User might want to:

U_1 Know in advance whether two similar queries $\mathbf{q}$ and $\mathbf{q}'$ yield the same k-ANN subset,

U_2 Explore efficiently a wider neighborhood of $\mathbf{q}$ by submitting few almost similar queries.

Note that this list of attacks is not exhaustive. It is worth repeating that the spirit of our work is not to prevent these threats absolutely. We enforce scalability first thanks to a moderately secure approach which yields a trade-off between the performance of the search and the feasibility of the attacks. In other words, instead of claiming that a threat is strictly impossible, we measure to which extend that threat is possible.

3. STATE OF THE ART

3.1 Secure NN search past approaches

We present some past approaches working for Euclidean distance based search. However, we omit solutions dealing with indexing (i.e. partitioning the vector space, see Sect. 1).

3.1.1 Homomorphic encryption

The Euclidean distance between the vectors of two parties can be computed without revealing them thanks to the homomorphic encryption primitive [1, 9, 7]. In a nutshell, the User sends an encrypted version of the query to the Server which, thanks to the homomorphism, sends back the encryption of the distance that the User deciphers.

This has two drawbacks. First, the Server knows $\mathcal{X}$ (threat S_1). If dishonest or if this database is stolen, exploitation of the data (threat S_2) is performed without the Owner's permission. On the other hand, threats S_3 and S_4 are impossible if the encryption is not broken. Threats U_1 and U_2 are not possible.

In practice, the computation of the Euclidean distance in the encrypted domain is slow and demands exchanging ciphers bigger in size than the vector. The search per se is exhaustive, running n times the protocol. There is no factorization between queries coming from two users since vectors must be processed by the public key of the User. This ‘secure’ k-NN takes in the order of 10 seconds to run the identification over a database of 320 entries [7, Tab. 3].

More general Secure Multiparty Computation (SMC) solutions have also been designed [7]. They rely on garbled circuits to securely evaluate a distance between two parties. Paper [4] introduces an efficient solution for Hamming distance based on Locality-Sensitive Hashing (LSH), which avoids the exhaustive search. However existing solutions for Euclidean distance-based search are still exhaustive and the database is stored in clear.

3.1.2 Hamming embedding

Another approach securely computes approximated distances. In the protocol of [3, Sect. IV-C], $\mathbf{x}_i$ is mapped to a binary representation $h(\mathbf{x}_i) \in \mathbb{B}^M$ $(\mathbb{B} = \{0, 1\})$ such that the Hamming distance between representations reflects the Euclidean distance between sufficiently close real vectors. This so-called Hamming embedding is parametrized by a matrix $\mathbf{A}$, a dither vector $\mathbf{w}$ and a quantization step Δ.

Since the Server needs these parameters to run the protocol, threat S_1 is possible according to [2] up to the quantization distortion. Threat S_2 is performed with the approximated distance. Nevertheless, threat S_3 is stopped because the Server never sees $h(\mathbf{q})$ in the clear ([3, Step 1]). S_4 is not feasible since $h(\mathbf{q})$ is semantically securely encrypted.

Threats U_1 and U_2 are not prevented in ([3, Step 3]) since the User controls the binary embedding of the query. Besides, the User sorts the distances ([3, Step 6]) and requires the metadata of the vectors it is interested in. The Server has no control on this selection.

The search is approximated (because based on Hamming distances) but exhaustive, requiring n homomorphic encryptions of the database representations *with the User public key* at the Server side ([3, Step 5]). This prevents the scalability of the search.

3.1.3 Attribute based encryption

Paper [8] builds a solution using attribute based encryption to avoid the last two drawbacks of 3.1.2. The User is able to decrypt the metadata t_i if and only if it knows a vector $\mathbf{q}$ such that $\|\mathbf{q}-\mathbf{x}_i\|^2 \leq \tau$ (vectors are here elements of $\mathbb{Z}^d$ and $\tau \in \mathbb{N}$). The enormous advantages follow:

- The database is composed of the metadata encrypted once for all with the Server public key,
- The Server does not store $\mathbf{x}_i$ or $h(\mathbf{x}_i)$.

Threats S_1, S_3, and S_4 are precluded. Threats U_1 and U_2 rarely occur for some specific setup. Yet, the Server which has the private key can unlock the ciphers (threat S_2).

However, the complexity is diabolic: the User must download the n encrypted metadata and perform τ decryptions (in interaction with the Server) per entry of the database to get the metadata t_i associated to the vectors $\mathbf{x}_i$ which are at most $\sqrt{\tau}$ away from $\mathbf{q}$ (if any).

3.2 An overview of PQ-codes

PQ-codes efficiently run k-ANN search at large scale [6].

3.2.1 Offline

The Owner has a database of vectors in $\mathbb{R}^d$: $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$. The vectors are split in into M subvectors of length ℓ. We assume $d = M\ell$ and denote $\mathbf{x}_i^{(m)} = (\mathbf{x}_i((m-1)\ell+1), \dots, \mathbf{x}_i(m\ell))$ the m-th subvector of $\mathbf{x}_i$. Denote $[a] = \{1, \cdots, a\}$ for any $a \in \mathbb{N}^*$. Then, $\forall m \in [M]$, the Owner runs a K-means over the subvectors in $\mathcal{X}^{(m)} = \{\mathbf{x}_i^{(m)}\}_{i\in[n]}$. It consists in randomly drawing K vectors in $\mathbb{R}^\ell$ and applying the Lloyd-Max algorithm until convergence. This ends up with a codebook of K centroids $\mathcal{C}^{(m)} = \{\mathbf{c}_i^{(m)}\}_{i\in[K]}$. This defines the m-th quantizer $Q^{(m)}(\cdots) : \mathbb{R}^\ell \to [K]$ as:

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

where $\|\cdot\|$ denotes the Euclidean distance. The K-means converges to a local minimum of the total reconstruction error distortion $\sum_{\mathbf{x}\in\mathcal{X}^{(m)}} \|\mathbf{x} - Q^{(m)}(\mathbf{x})\|^2$. To shorten this preparation time, the Owner applies it on a training set which is a random subset of $\mathcal{X}^{(m)}$. The results of the K-means depends on this subset, the initial random sampling, and the number of iterations. We define the global quantizer $Q(\cdot) : \mathbb{R}^d \to [K]^M$ as the product quantizer $Q^{(1)} \times \ldots \times Q^{(m)}$:

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

We denote by $Q^{-1}(\cdot) : [K]^M \to \mathbb{R}^d$ the operator mapping a sequence of indices to the concatenation of centroids:

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

The Owner sends the Server the database $\mathcal{Q} = \{Q(\mathbf{x}_i)\}_{i\in[n]}$ (i.e. $h(\cdot) = Q(\cdot)$) and the set of M codebooks $\mathcal{C} = \{\mathcal{C}^{(m)}\}_{m\in[M]}$. The role of the Owner stops here.

3.2.2 Online: the symmetric search

The Server pre-computes the distances between centroids of the same codebook:

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

The matrix d_s will be used as a lookup table.

Online, when receiving a query $\mathbf{q}$ from the User, the Server first computes $Q(\mathbf{q})$. It proceeds the k-ANN search based on the approximated square distance

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

instead of the true square distance $\|\mathbf{q} - \mathbf{x}_i\|^2$. This is efficiently done thanks to the lookup table:

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

The min-heap algorithm returns the indices $(i_1, \dots, i_k)$ yielding the k smallest approximate distances. The Server sends the metadata $(t_{i_1}, \dots, t_{i_k})$ associated to these k vectors.

There exists a variant of PQ-codes, so-called asymmetric search, which is not used in the paper.

4. SLOWLY RISING THE WALLS

The goal of this section is to underline the relationships between the threats listed in Sect. 2.2 and the key elements of PQ-codes, which are the centroids codebook $\mathcal{C}$ and the distance table d_s. We start our analysis with the original PQ-codes as presented in Sect. 3.2.

4.1 Scenario 1: original PQ-codes

First, the Server cannot reconstruct $\mathbf{x}_i$, but only an estimation $\hat{\mathbf{x}_i} = Q^{(-1)}(Q(\mathbf{x}_i))$ because it has the indices $\mathcal{Q}$ and the centroids of $\mathcal{C}$ (threat S_1). Second, the Server can run k-ANN searches without the Owner's permission, e.g. with the purpose of clustering the vectors of $\mathcal{X}$ (threat S_2). Obviously, PQ-codes are not compliant with privacy because the User sends his query $\mathbf{q}$ in the clear to the Server (threats S_3 and S_4). On the other hand, this renders the User harmless (threats U_1 and U_2 are void).

4.2 Scenario 2: confiscating the codebook

Suppose that we succeed to make the query quantization at the User side. Then, the Server no longer needs $\mathcal{C}$.

Having d_s, the Server knows the $K(K-1)/2$ distances between the K centroids of $\mathcal{C}^{(m)}$, $\forall m \in [M]$. Since K is usually much bigger than the subspace dimension ℓ, the Server can construct a constellation of K points sharing the same inter-distances. This does not fully disclose the codebook $\mathcal{C}$, but up to an ambiguity which is an isometry of $\mathbb{R}^\ell$, i.e. a transformation of the space that preserves distances (say a rotation followed by a translation).

This ambiguity plus the quantization loss is sufficient for preventing an accurate reconstruction of the database vectors from $\mathcal{Q}$ (threat S_1) and the query vector from $Q(\mathbf{q})$ (threat S_3). The Server cannot query alone, but it can cluster the database vectors according to their approximated distances $\hat{D}(\mathbf{x}_i, \mathbf{x}_j)$ thanks to the lookup table d_s (threat

S_2). The Server can detect almost similar queries $\mathbf{q}$ and $\mathbf{q}'$ by computing $\hat{D}(\mathbf{q}, \mathbf{q}')$ (threat S_4).

To perform the quantization of the query, The User is being given the centroids. Now, he knows in advance that two queries $\mathbf{q}$ and $\mathbf{q}'$ yield the same k-ANN if $Q(\mathbf{q}) = Q(\mathbf{q}')$ (threat U_1). He can also adapt his query: forging a query $\mathbf{q}'$ which equals $\mathbf{q}$ except for one subvector pertaining to a different Voronoi cell will yield another set of k-ANN vectors. In other words, he can explore a wider neighborhood of $\mathbf{q}$ more efficiently (i.e. with less queries - threat U_2).

4.3 Scenario 3: confiscating the lookup table

Suppose now that the Server knows neither $\mathcal{C}$ nor d_s. It does not have the centroids, which prevents vector reconstruction, be it from the database (threat S_1) or the query (threat S_3). It is missing d_s to compute approximated distances between entries of $\mathcal{Q}$. Yet, it can still infer database vector neighborhood by forging the lookup table:

$$d_p(i, j, m) = 1 - \delta_{i,j}, \ \forall (i, j, m) \in [K] \times [K] \times [M], \quad (7)$$

where $\delta_{i,j}$ is the Kronecker function ($= 1$ if $i = j$, 0 otherwise). This method provides a very crude approximation of nearest neighbors (see Fig. 3 blue dotted line). In other words, threat S_2 seems to be barely feasible. However, the following section provides a working implementation of this scenario but this particular threat is not totally precluded.

5. OUR PROPOSAL

The previous section demonstrated that the Server can hijack information and threaten the entire system. We propose in this section several mechanisms making the job of the curious Server more difficult for threatening the security and privacy of k-ANN searches with PQ-codes. The main idea to enforce the above-mentioned Scenario 3 is the introduction of two quantizers.

5.1 The algorithm

The Owner creates offline $\mathcal{C}_S$, a set of M codebooks of K_S centroids each. This defines the product quantizer $Q_S(\cdot)$ used to create the database $\mathcal{Q} = \{Q_S(\mathbf{x}_i)\}_{i=1}^n$ given to the Server. Only the Owner knows $\mathcal{C}_S$.

The Owner also creates $\mathcal{C}_U$, a set of M codebooks of K_U centroids each, defining $Q_U(\cdot)$. $\mathcal{C}_U$ will be sent to the User to quantize $\mathbf{q}$. The Owner also computes the square distances:

$$d_{us}(i, j, m) = \|\mathbf{c}_{U,i}^{(m)} - \mathbf{c}_{S,j}^{(m)}\|^2, \ \forall (i, j, m) \in [K_U] \times [K_S] \times [M], \quad (8)$$

and sends this lookup table to the Server.

Online, the User gets $\mathcal{C}_U$, sends $Q_U(\mathbf{q})$ to the Server which performs the ANN search with d_{us}. Note that the quantizers may not have the same number of centroids per subspace. It is important to have a reasonable K_S because the memory footprint of $\mathcal{Q}$ at the Server side is $nM \log_2 K_S$ bits. A bigger K_U improves the quality of the approximative search, while payload of the transmission between the User and the Server, i.e. $M \log_2 K_U$, slightly increases.

5.2 Threat analysis

5.2.1 *Vector reconstruction*

As claimed in Sect. 4.2, the Server cannot reconstruct database vectors (threat S_1) because it misses the knowledge of $\mathcal{C}_S$. The same is true for query vectors (threat S_3) because it does not have $\mathcal{C}_U$. Note that this holds as long as there is no collusion between a User and the Server, or as long as the Server cannot usurp the role of the User. These two cases are excluded in the ‘honest but curious’ model.

5.2.2 *Similar queries detection*

The Server obviously spots similar queries $\mathbf{q}$ and $\mathbf{q}'$ where $Q_U(\mathbf{q}) \approx Q_U(\mathbf{q}')$ (threat S_4). However, it has difficulty in gauging how much different are these two queries because it is missing the distance table between centroids of $\mathcal{C}_U$.

5.2.3 *Clustering the database*

For a given entry, the Server knows $Q_S(\mathbf{x}_i)$ whereas it would need $Q_U(\mathbf{x}_i)$ to compute the approximated distances against the other entries of $\mathcal{Q}$ thanks to d_{us}. This is the reason why we measure the averaged mutual information between results of a quantization onto $\mathcal{C}_U^{(m)}$ and $\mathcal{C}_S^{(m)}$:

$$I(Q_S; Q_U) = M^{-1} \sum_{m=1}^{M} I(Q_S^{(m)}(\mathbf{X}_i^{(m)}); Q_U^{(m)}(\mathbf{X}_i^{(m)})). \quad (9)$$

The computation of this quantity is easy since we deal with discrete random variables.

Another angle of attack is to estimate d_s defined in (4). Eq. (7) was a first attempt, but the Server can do much better thanks to d_{us} defined in (8). The idea is simple: if $d_{us}(i, j, m)$ is close to zero, it means that $\mathbf{c}_{U,i}^{(m)}$ is close to $\mathbf{c}_{S,j}^{(m)}$, therefore the distance $d_{us}(i, k, m)$ should be a good estimation of $d_s(j, k, m)$. The estimation goes as follows:

$$\begin{aligned} \hat{d}_s(j, k, m) &= (d_{us}(I(j), k, m) + d_{us}(I(k), j, m))/2, \\ \text{with } I(j) &\triangleq \arg \min_{i \in [K_U]} d_{us}(i, j, m). \end{aligned} \quad (10)$$

The performances of the k-ANN search with this estimated distance table are slightly lower than with d_{us} (Fig. 3). This means that threat S_2 cannot be prevented. Note that our approach is close to one-way private search [5] where only the User’s data are sensitive.

5.2.4 *Threats from the User*

Knowing $\mathcal{C}_U$ and thus the Voronoi cells associated to each subquantizer, the User knows which queries in the space will yield the same k-ANN (threat U_1): it holds for any $(\mathbf{q}, \mathbf{q}')$ such that $Q_U(\mathbf{q}) = Q_U(\mathbf{q}')$. In the same way, he can efficiently explore portion of the space by submitting queries almost identically quantized (threat U_2).

If these latter threats are annoying for the targeted application, then a secure distance computation protocol (as in Sect. 3.1.1) is a solution. The Server generates (sk_S, pk_S) for an additive homomorphic crypto-system $e(\cdot)$. The owner encrypts the $e(\mathbf{c}_{U,i}^{(m)}, pk_S)$ and $e(\|\mathbf{c}_{U,i}^{(m)}\|^2, pk_S)$ offline. These ciphers are privately sent to the User who computes and sends $e(\|\mathbf{q}^{(m)} - \mathbf{c}_{U,i}^{(m)}\|^2, pk_S)$ back to the Server. The Server decrypts and computes $Q_U(\mathbf{q})$ knowing neither $\mathbf{q}$ nor $\mathcal{C}_U$. The User no longer sees $Q_U(\mathbf{q})$. The User together with the Server have to compute in the encrypted domain $M.K_U$ distances, which is much fewer than n as proposed in 3.1.1. These secure computations last longer than the ANN search, so that the runtime is dominated by this constant duration: this does not spoil the scalability of PQ-codes.

We can even ensure that the Server learns only the value of $Q_U(\mathbf{q})$ and nothing else. To this aim, the server computes for each $m \in [M]$, $Q_U^{(m)}(\mathbf{q}^{(m)})$, i.e. the argmin of the encrypted distances $e(D_1, pk_S), \ldots, e(D_{K_U}, pk_S)$ with $D_i = \|\mathbf{q}^{(m)} - \mathbf{c}_{U,i}^{(m)}\|^2$, interactively without decrypting the distances. This prevents the Server from learning the intermediate results. First, the User encrypts the distances through El Gamal encryption $E[\cdot]$ with its public key pk_U associated to its secret key sk_U and sends the Server the results $E[e(D_1, pk_S), pk_U], \ldots, E[e(D_{K_U}, pk_S), pk_U]$. The server permutes these ciphers to randomize their order and computes, thanks to the multiplicative homomorphism of El Gamal,

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

with a random $\alpha > 0$. This in turn, thanks to the additive homomorphism of $e(\cdot)$, leads to:

$$E[e(\alpha.D_{i1}, pk_S), pk_U], \ldots, E[e(\alpha.D_{iK_U}, pk_S), pk_U]. \quad (12)$$

The role of α is to blind the ciphers such that the User cannot guess the permutation. The Server sends back the data to the User who decrypts those but without being able to retrieve the original order of the data.

Then, the User and the Server execute an interactive sorting algorithm by comparing the distances in the encrypted domain following the principle of Yao's millionaire problem. The secure comparison of two encrypted data $e(x, pk_S)$ and $e(y, pk_S)$ is made as follows: let R a big random element and R' significantly smaller than R, the User computes $e(R(x-y) - R', pk_S)$ thanks to the homomorphic property. The Server decrypts this message and if it gives a positive value, it decides that $x > y$. This enables the Server to determine the index of the minimum distance between the K_U distances by executing $K_U - 1$ successive secure comparisons with the User. As the order is only known by the Server, the Server obtains $Q_U^{(m)}(\mathbf{q}^{(m)})$ at the end whereas the User will not learn the result. Doing so for all m leads to $Q_U(\mathbf{q})$. This secure computation of $Q_U(\mathbf{q})$ has the advantage that the Server and the User learns the minimum level of details.

6. EXPERIMENTAL BODY

Our experiments are performed on the *ANN_SIFT1M* local SIFT descriptors database introduced in [6]. Note that the ANN_SIFT1M database consists of (i) $1,000,000$ base vectors of dimension $d = 128$, (ii) $100,000$ training vectors for running the K-means, and (iii) $10,000$ query vectors and a ground truth file which contains, for each query, the identifiers of its nearest neighbors ordered by increasing distance.

6.1 Quality of the search

PQ-codes performs a k-ANN search, meaning that the returned NN are not necessary the true ones. To gauge the quality of the output, the recall at rank $R \leq k$, denoted by 'r-recall@R' is measured. This is the proportion of query vectors for which the r-NN are ranked in the first R returned vectors. As usually done in ANN search papers, we focus on the 1-recall@R. Fig. 1 shows the 1-recall@R in percentage. On the server side, PQ-codes are performed with $M = 16$, $l = 8$, $K_S = 256$ and $N_i = 50$, the number of iterations of K-means process. The dashed line shows the performances of the original PQ-codes. In brief, the search returns almost surely the NN for R = 100. We increase the number of centroids for the quantizer Q_U (from $K_U = 64$ to 4096). This gives a better quality of search when $K_U > K_S$.

Usually, the number of centroids is a power of two, so that the memory footprint of the database is $nM \log_2 K_S$ bits. This is a very compact representation of $\mathcal{X}$. The time response is linear with nM. In our setup with $n = 10^6$, $M = 16$, $K_S = 256$, the database $\mathcal{Q}$ occupies 16MB. Once $Q_U(\mathbf{q})$ is computed, one approximated search is completed within 30 ms (Core i7 platform, single threaded). Parameter K_U has almost no impact on the time response, provided d_{us} can fit in memory.

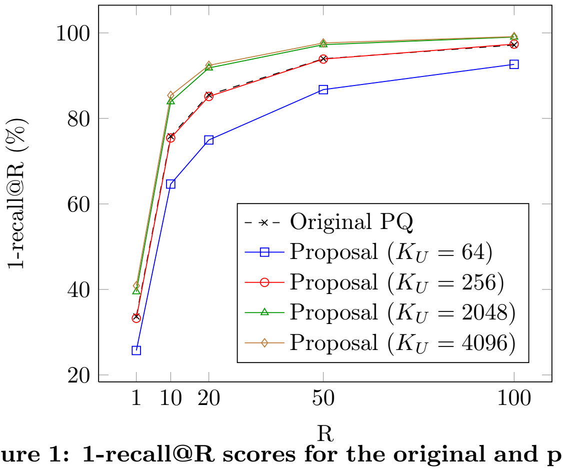

Figure 1: 1-recall@R scores for the original and proposed version of PQ-codes: $M = 16$, $l = 8$, $K_S = 256$, $N_i = 50$.

6.2 Threat S_2

The curious Server has two possibilities for running k-ANN searches within the database. A first attempt is to use d_{us}, but the database vectors are improper because they are not quantizations onto $\mathcal{C}_U$. We measure the average amount of information per quantizer $I(Q_S; Q_U)$ (see (9)) that the curious Server is missing for using d_{us}. Fig. 2 shows this amount w.r.t the number of iterations of the K-means algorithm, computed on the ANN_SIFT1M training database. The dashed line shows the entropy of $Q_S^{(m)}(\mathbf{X}^{(m)})$ when the quantization of a vector is equiprobably distributed, i.e. $\log_2(K_S)$. The black line (cross markers) shows the estimation of this entropy, which is smaller. This is due to the fact that the goal of the K-means is to minimize the mean square error, not to assure the equiprobability distribution. $I(Q_S; Q_U)$ increases with K_U, but do not reach the value of the entropy. Therefore, the curious Server is missing an amount of information which is in the order of $M.(H(Q_S) - I(Q_S; Q_U))$ bits per entry of the database to use the table d_{us}. The bigger is K_U, the bigger is the information leakage while increasing the quality of search (see Fig. 1). We can see here the price to pay for more security.

Note that for a few iterations of the K-means process, the distribution of the centroids is more random, the gap is bigger, and so the system more secure against this attack. The reconstruction error distortion is not optimal, but we have noticed that this number of iterations has no impact on the quality of search provided it is ≥ 3.

In a second attack, the curious Server either uses d_p of (7), or estimates the missing distance table d_s via (10). Fig. 3 shows the 1-recall@R scores when the Server utilizes (i) the lookup table d_{us}, (ii) the Kronecker lookup table d_p (7) and (iii) the estimated $\hat{d}_s$ (10) for different K_U (number of iterations of K-means process is 3).

The Kronecker version yields a recall@R below 30% for

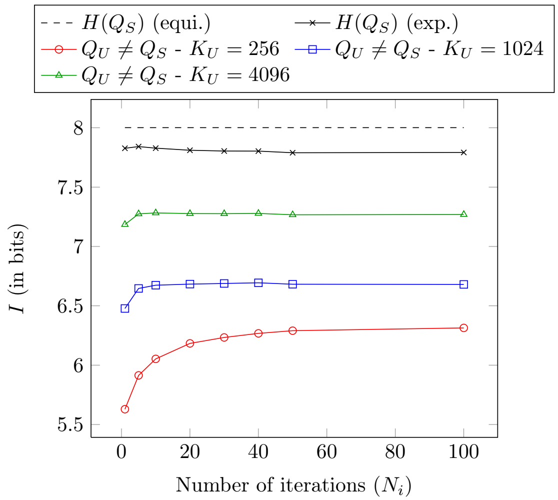

Figure 2: Empirical mutual informations between the two quantizers Q_S and Q_U w.r.t the number of iterations of the K-means with $M = 16$, $K_S = 256$.

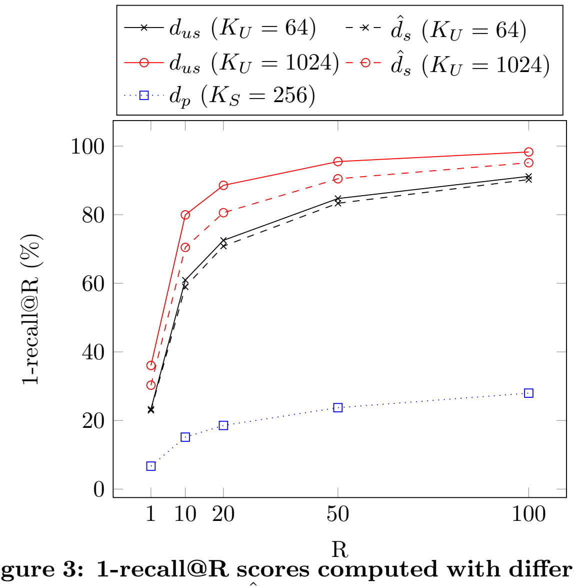

Figure 3: 1-recall@R scores computed with different lookup tables $\{d_{us}, d_p, \hat{d}_s\}$ with $M = 16$ and $K_S = 256$.

any rank $R \leq 100$. The quality of search is too weak for a possible clustering of the database. The attack based on the estimation $\hat{d}_s$ works much better. A large K_U improves the accuracy of the estimation, and the performances are almost equal to the original PQ-codes with $K = K_S$. However, increasing K_U to some extend also improves the quality of search for the User (because the query is more finely quantized by Q_U). At some point, both curves do not evolve, and rising K_U even more just increases computation time and bandwidth for nothing.

6.3 Threats U_1 and U_2

Sect. 5.2.4 prevents these threats but at a huge cost in terms of computation and bandwidth. Let us roughly evaluate the bandwidth first (figures are for $K_S = 256$). The User needs the encrypted centroids and their norm, i.e. around $M(\ell+1)K_U \times 2048$ bits (10MB). This can be factorized over several queries. The User sends distances encrypted with El Gamal, i.e. $MK_U \times 4096$ bits (2MB). The Server sends back these ciphers, i.e. same amount. For the Yao protocol, the User sends $MK_U \times 2048$ bits (1MB) to the Server. As for the computation times, the User makes $O(MK_U(\ell+3))$ exponentiations (∼50 sec. on a regular PC) and the Server $O(2MK_U)$ (8 sec. on a regular PC).

7. CONCLUSION

The advantages of this proposal are that (i) the database at the Server side is fixed, (ii) there is no loss w.r.t. the quality of the search, (iii) the complexity and bandwidth bottleneck depends on K_U but not on n. The preliminary protocol is a protection against threats from Users. The drawback of our proposal is that a Server can search within the database (for clustering e.g.) with a slight loss of accuracy compared to quality of search provided to the User. Note that none of past approaches protect against this threat.

8. ACKNOWLEDGMENTS

This work was supported in part by the National French project ANR-12-CORD-0014 SecuLar.

9. REFERENCES

[1] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, and A. Piva. Privacy-preserving fingercode authentication. In *Proceedings of the 12th ACM workshop on Multimedia and security*, pages 231–240, NY, USA, 2010. ACM.

[2] P. Boufounos. Universal rate-efficient scalar quantization. *Information Theory, IEEE Transactions on*, 58(3):1861–1872, 2012.

[3] P. Boufounos and S. Rane. Secure binary embeddings for privacy preserving nearest neighbors. In *Information Forensics and Security (WIFS), IEEE International Workshop on*, pages 1–6, 2011.

[4] J. Bringer, M. Favre, H. Chabanne, and A. Patey. Faster secure computation for biometric identification using filtering. In *Biometrics (ICB), 5th IAPR International Conference on*, pages 257–264, 2012.

[5] G. Fanti, M. Finiasz, and K. Ramchandran. One-Way Private Media Search on Public Databases: The Role of Signal Processing. *IEEE Signal Processing Magazine*, 30(2):53–61, 2013.

[6] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):117–128, 2011.

[7] R. Lagendijk, Z. Erkin, and 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.

[8] S. Rane and W. Sun. An attribute-based framework for privacy preserving image querying. In *Image Processing (ICIP), 19th IEEE International Conference on*, pages 2649–2652, 2012.

[9] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *Information, Security and Cryptology - ICISC 2009*, volume 5984 of *Lecture Notes in Computer Science*, pages 229–244. Springer Berlin Heidelberg, 2010.