Privacy-Preserving k-NN for Small and Large Data Sets

Author

Amirbekyan, A, Estivill-Castro, V

Published

2007

Conference Title

Proceedings - IEEE International Conference on Data Mining, ICDM

DOI

10.1109/ICDMW.2007.67

Privacy-Preserving k-NN for Small and Large Data Sets

Artak Amirbekyan
Griffith University
Meadowbrook QLD 4131, Australia
A.Amirbekyan@gu.edu.au

Vladimir Estivill-Castro
Griffith University
Meadowbrook QLD 4131, Australia
V.Estivill-Castro@gu.edu.au

Abstract

It is not surprising that there is strong interest in k-NN queries to enable clustering, classification and outlier-detection tasks. However, previous approaches to privacy-preserving k-NN are costly and can only be realistically applied to small data sets. We provide efficient solutions for k-NN queries queries for vertically partitioned data. We provide the first solution for the L_∞ (or Chessboard) metric as well as detailed privacy-preserving computation of all other Minkowski metrics. We enable privacy-preserving L_∞ by providing a solution to the Yao's Millionaire Problem with more than two parties. This is based on a new and practical solution to Yao's Millionaire with shares. We also provide privacy-preserving algorithms for combinations of local metrics into a global that handles the large dimensionality and diversity of attributes common in vertically partitioned data.

1 Introduction

Data mining has been identified as one of the most useful tools for the fight on crime [19]. But the information needed resides with many different data holders, that may mutually do not trust each other, or between parties that have conflicts of interest. All parties are aware of the benefits brought by collaboration, but none of them wants the others or any third party to learn their private data. For this kind of collaboration, data privacy becomes extremely important. For most data mining algorithms, the data is encoded as vectors in high dimensional space. For these algorithms, a measurement of similarity (or dissimilarity) is necessary and many times fundamental for their operation. In large databases, content-based retrieval under the vector model must typically be implemented as k-nearest-neighbour (k-NN) queries, whose result consists of the k items whose features most closely resemble those of the query vector according to the similarity measure. Nearest neighbor (NN) queries [20] have been extensively studied in the past [3, 5, 7]. The k-NN queries enable Local Outlier Detection [21], Shared NN Clustering [21] and k-NN Classification [2, 18, 21]. These wide variety of data mining tasks, for which k-NN queries is a fundamental operation, has recently prompted approaches to privacy preserving k-NN [2, 18, 21, 23, 24]. However, these approaches do have some serious shortcomings. For example, the suite of privacy preserving algorithms [24] to create a privacy preserving version of Fagin's *A0* algorithm [10] proved costly despite the authors argued that disclosure of some additional information (the union of all items in a set required to get k intersecting items) was necessary for reasonable efficiently. Other limitations have commonly been the need to compute all pairs of distances [23], to have the query-point public [18], or to deal with horizontally partitioned data [2, 21]. To manage very large data sets, we use a privacy-preserving $SASH$. With a theoretical analysis we also illustrate the efficiency of our approach with an empirical evaluation.

2 Private collaborations

We study collaboration between several parties that wish to compute a function (say f) of their collective databases. Each wants the others to find as little as possible of their own private data. In vertically partitioned data, every record in the collective database is an attribute-value vector, where every party owns some part (a number of attributes) from that vector. For simplicity, we can identify each domain with one party if we assume existance of virtual parties [1], so the dimension m of the records is also used as the number or parties. This simplifies the notation in the algorithms and communication between two virtual parties of the same party does not need to occur. Our approach is based on the theory developed under the name of "secure multiparty computation" (SMC) [12]. Yao's Millionaires Problem [27] provides the origin for SMC. Secure multi-party computation under the semi-honest model [12] has regularly been used for privacy-preserving data mining [8, 9, 22]. Here, we work under the semi-honest model as well, which means all parties will follow the protocol since all are interested on the results. However, all parties can use all the information collected during the protocol to attempt to discover the private data or some private values from another party. The SMC literature has a general solution for all polynomially bound computations [13]. However, the general solution requires f to be explicitly represented as a Boolean circuit of polynomial size. Even if represented as a circuit of polynomial size, the inputs must be very small for the circuit to be practical. This means, the sub-task that uses this result must be used on very small inputs, a constraint difficult to meet in data mining applications.

[1] If Alice(the first party) holds 5 attributes, then there are 5 virtual parties, one for each columns.

0-7695-3019-2/07 $25.00 © 2007 IEEE
DOI 10.1109/ICDMW.2007.67

2.1 Yao's two millionaires problem - solution with shares

One advantage of the "secret shares" theoretical result is that one can easily decompose the result $f(\vec{x}, \vec{y})$ into a share s_A for Alice and a share s_B for Bob, so that $s_A + s_B = f(\vec{x}, \vec{y})$, but neither party can find $f(\vec{x}, \vec{y})$ from their share. This allows to use a protocol for one task (like Yao's comparison of two values) in a larger protocol (e.g. sorting). We present here a practical solution to Yao's millionaires problem, but provide the output in secret shares. Recall that here Alice holds a and Bob holds b, but then, after the protocol, they do not share knowledge of the output ($a > b$?), but the output for Alice is r_a and for Bob r_b, where $r_a + r_b = 1$ if $a > b$ and $r_a + r_b = 0$ if $a \leq b$. There are several algorithms [18, 21, 23] that invoke a subroutine for Yao's comparison with secret shares, and all of them rely on the circuit evaluation generic "shares" theoretical solution [12]. We present here a practical and inexpensive solution that we apply in our algorithms. It uses a third untrusted party (also commonly used in the privacy-preserving literature [9]). Checking whether $a > b$ is the same as checking whether $a + (-b) > 0$. In our protocol, we will use an untrusted non-colluding third party (also called semi-trusted [6]).

Protocol 1 1. *The third party generates a random number R_a and sends R_a to Alice, who then generates random number R, where $R \in \Re \backslash \{0\}$ and sends $(R, aR + R_a)$ to Bob.*

2. *Bob adds $-bR$, and sends $(a-b)R + R_a$ to the third party, who subtracts R_a and checks whether $(a - b)R > 0$.*

3. *The third party then generates two pairs of values (r_a^0, r_b^0) and (r_a^1, r_b^1), where $r_a^0 + r_b^0 = 0$ and $r_a^1 + r_b^1 = 1$. If $(a-b)R < 0$, then the third party sends (r_a^0, r_a^1) to Alice and (r_b^0, r_b^1) to Bob. If $(a - b)R \geq 0$, then the third party sends (r_a^1, r_a^0) to Alice and (r_b^1, r_b^0) to Bob.*

4. *If $sign(R) = 1$ (i.e. $R > 0$) Alice and Bob use as their shares the first value in the pair received from the third party. While if $sign(R) = -1$, each picks as their share the second number in the pair received from the third party.*

Here Alice obtains R, R_a and the set $\{r_a^1, r_a^0\}$. However, has no way of telling which one is which in the set $\{r_a^1, r_a^0\}$. Bob obtains R, the set $\{r_b^1, r_b^0\}$, and $a \cdot R + R_a$. Again, Bob can not tell which one is which on the set $\{r_b^1, r_b^0\}$. The third party gets $R_a, r_a^1, r_a^0, r_b^1, r_b^0$, and $(a-b) \cdot R$. Note that, the third party does not know the sign of $a - b$, since $sign(R) \in \{-1, 1\}$. A small concern is that in the case $a = b$, the third party learns that $a = b$, although it does not learn anything else (the values a and b remain unaccessible to the third party). It is possible to increase the uncertainty in the third party by Alice and Bob using the third party with additional dummy Yao's tests. A security proof by simulation [12] is obvious, Bob just needs to get a random value for $a \cdot R + R_a$ and in polynomial time adds $-b \cdot R$ wth output given by the third party. Alice's simulation is even more trivial since she receives nothing from Bob. The output is also from the third party. We have implemented the sing-based solution presented above. Implementation requires some adaptation, since the value $R \in \Re \backslash \{0\}$ generated by Alice cannot be any non-zero real. However, all implementations of a solution to Yao's Millionaires problem assume that the values of a for Alice and b for Bob are in a large but bounded interval; that is $a, b \in [0, M]$ where M is very large (and known to both Alice and Bob). Even those solutions that do not provide the answer with shares require this and typically assume further that a and b are integers in a very large field. Therefore, our implementation of the sign-based solution also assumes that a and b are integers with $a, b \in [0, M]$, and M is known to the implementer. Since Bob will learn R and $aR + R_a$, the value R_a produced by the commodity server must mask aR (otherwise Bob may learn some bits about a). Since aR can have as many bits as $\log_2 M + \log_2 R$, we typically let Alice chose R so that $|R| > M$ and the third party chooses $R_a > 2M$. For example, Alice can chose R uniformly in $[-2M, M) \cup (M, 2M]$. After this adjustment, the implementation does not reveal information to Alice or Bob about each others value. However, the third party does learn $(a - b)R$ and because $|R|$ is bounded (in fact, the distribution of $|R|$ will be know to the third party), this party learns approximations to $|a - b|$. That is, the third party will not learn whom between Alice and Bob holds the larger value, but will gain an idea on the gap that exists between the two. We regard this leak of information in the sign-based protocol as innocuous given the estimates of $|a - b|$ have relative error and the other advantages it provides. First, it is the first implementable protocol for Yao's millionaires problem with shares (we have a C++ implementation over sockets). Second, it is far more efficient that other solutions to Yao's millionaires problem, even without shares. Cachin's solution [6] [2] is linear on $\log_2(a + b)$; however, it also requires a semi-trusted party and very heavy cryptographic machinery. A solution that has been demonstrated to be efficient enough for ETHERNET networks [17] requires quadratic time and quadratic number of messages on $\log_2(a + b)$ and also as many oblivious transfers as $\log_2(a+b)$. Other practical protocols [4] also require $O(log_2(a + b))$ rounds of oblivious transfer. Oblivious transfer implementation usually requires at least two messages with a key each. The sign-based protocol requires three message with a $\log_2(a+b)$ (one from the third party to Alice, one from Alice to Bob and one from Bob to the third party). The last round of messages have constant size 2 bits. So we have linear complexity on the size of the message (with a constant value 2) and constant number of messages. This analysis is so overwhelming clear in favor of the sign-based protocol that we feel direct comparison to any other implementation of a solution to Yao's millionaires unnecessary.

3 Privacy-Preserving Metrics

Obviously, if all parties know the r-th $Minkowski$ distance $M(\vec{p}, \vec{q})$ between two points $\vec{p}$ and $\vec{q}$ in the database, they will also know the value $[M(\vec{p}, \vec{q})]^r$ by each raising the $Minkowski$ distance to the r-th power. Conversely, if the parties find $[M(\vec{p}, \vec{q})]^r$, they can take r-th roots and find the desired distance value. Since the i-th party knows a range of the attributes of the vectors $\vec{p}$ and $\vec{q}$, we have

$$[M(\vec{p}, \vec{q})]^r = \sum_{i=1}^{m} \sum_{\substack{\text{attr. known} \\ \text{to party } i}} \left(\begin{smallmatrix} j\text{-th attr. in } \vec{p} \\ \text{owned by } i \end{smallmatrix} - \begin{smallmatrix} j\text{-th attr. in } \vec{q} \\ \text{owned by } i \end{smallmatrix} \right)^r .$$

[2] We let $\log_2(a + b)$ denote the number of bits of $a + b$.

Letting v_i be the r-th *Minkowski* distance of those attributes known to the i-party, then $[M(\vec{p},\vec{q})]^r = v_1^r + \cdots + v_m^r$, and the problem reduces to finding the value of the sum of values distributed among m parties (each contributes the knowledge of the r-th *Minkowski* distance raised to the power r in the projection that they own). This can be computed by circular accumulator sum (also called secure sum [25]). Note that if we halt the circular sum (at the second party Bob), then the distance value will be shared between Bob and Alice. This means Alice holds $a = v_1^r - R$ and Bob holds $b = (v_2^r + \cdots + v_m^r + R)$ where R is a random number produced by Alice and $[M(\vec{p},\vec{q})]^r = a + b$. In some applications (in particular, k-NN queries) the calculation of the metric is an intermediate step. Although distance values may be considered innocuous information, it is more acceptable to use shares as this adheres to the ideal principle of SMC where parties learn only what is implied by the final output. In fact, we can use encryption modulo a field F so that the shares s_a of Alice and s_b of Bob are such that $s_a + s_b = [M(\vec{p},\vec{q})]^r \mod F$. The *Minkowski* distance protocol with shares is trivial in the case $m = 2$ parties, since in this case, each party uses its projected metric value as its share and they exchange no messages at all. Thus, $m > 3$ parties use the *Minkowski* distance protocol, no party learns other parties' data; ie, no attribute in a feature vector is revealed to another party. If the protocol is the shares distance version, no party learns any information even if $m = 2$. This is clear, beacuse the protocol calculates the distance between $\vec{p} = (p_1, \cdots, p_m)$ and $\vec{q} = (q_1, \cdots, q_m)$. During the calculation each party P_l ($l = 2, \cdots, m$) obtains $S_l = (p_{l+1} - q_{l+1})^r + \cdots + (p_m - q_m)^r + R \bmod F$. Thus, for party P_l it is impossible to learn any value from $(p_{l+1} - q_{l+1})^r$ up to $(p_m - q_m)^r$ and $(p_1 - q_1)^r$. In the case of the first party ($l = 1$), it obtains the actual distance $[M(\vec{p},\vec{q})]^r$ by subtracting R, adding its projection and taking $r - th$ root from the sum. Here, because several terms are involved in the sum, the 1^{st} party cannot learn any attribute p_i or q_i, where $i = 2, \cdots, m$. The case for shares follows by the discussion above. Again, a security proof by simulation [12] is obvious, since each party takes as input a simulated random number from the previous party and proceeds in polynomial time. The overall result is obtained because the first party can use in the simulation the overall result. The L_∞ metric (some prefer the name "Chessboard distance") is defined as $dist(\vec{x},\vec{y}) = \max(|v_1^x + (-v_1^y)|, ..., |v_m^x + (-v_m^y)|)$ for vectors $\vec{x} = (v_1^x, \cdots, v_m^x)^T$ and $\vec{y} = (v_1^y, \cdots, v_m^y)^T$. Note that if $\vec{x}$ and $\vec{y}$ are owned by several parties on vertically partitioned data, each party can identify the largest absolute difference $v_i = |v_i^x + (-v_i^y)|$ in its projection. So the problem reduces to which of the m parties has the largest value.

Protocol 2 *Find maximum value with shares.*

1. *The protocol starts with each party comparing its value to every other party.*[3] *So Alice (the first party) compares her value with all others, then puts the sum of her shares as the first component in her shares* column *vector* $\mathbf{P_1}$. *All other parties put their shares, that come from comparisons with Alice, again as the first component of their shares* column *vector. This creates a distributed* row *vector.*

2. *Bob (the second party) compares his value with all others*[4]*, and puts the sum of his shares as the second component in his shares* column *vector* $\mathbf{P_2}$. *All other parties put their shares, that come from these comparisons with Bob as the second component of their shares* column *vector. This creates a distributed* row *vector.*

3. *The protocol continues until each party's value is compared with all others and all column vectors are obtained.*
This provides the information shown in the Table 1 where C_{ij}^i

$\mathbf{P_1}$	$\mathbf{P_2}$	$\cdots$	$\mathbf{P_m}$
$\sum_{j=2,..,m} C_{1j}^1$	C_{12}^2	$\cdots$	C_{1m}^m
C_{21}^1	$\sum_{j=1,..,m}^{j\neq 2} C_{2j}^2$	$\cdots$	C_{2m}^m
.	.	.	.
C_{m1}^1	C_{m2}^2	. . .	$\sum_{j=1,..,m-1} C_{mj}^m$

Table 1. Shares after all comparisons.

belongs to P_i, C_{ij}^j *belongs to* P_j, *and* $C_{ij}^i + C_{ij}^j = 1$ *if* $v_i > v_j$, *when* $v_i \leq v_j$ *then* $C_{ij}^i + C_{ij}^j = 0$ *Note that now, each column is owned by one party only; therefore we can treat them as the vectors distributed, one to each party. Moreover, the sum of the elements in the* i*-th row is* $\sum_{j=1,..,m}^{j\neq i} C_{ij}^i + \sum_{j=1,..,m}^{j\neq i} C_{ij}^j$, *will show us exactly how many* v_j *were smaller than* v_i. *That is, the ranking of the distance projection owned by the* i*-th party. The problem now reduces to finding the ID (identifier) of the maximum in a sum of vectors*[5]. *This can be performed using the "Maximum Value in the Sum of Vectors" protocol [2] where no party learns anything except the ID of the entry holding the maximum value. Once the party with the highest ranking is known, this party can broadcast the maximum value (the value of the metric).*

If a version with shares is needed, the party P holding the maximum value M can generate a random number $s_P = R$, so that $s = M - R$ is made public to another party. Then, these two parties will hold values so that $s_p + s = M \mod F$. This algorithm is as secure as our comparison protocol, because parties do not learn winners or losers of the comparisons since all encoded in distributed shares. The *Minkowski* metric may not be powerful enough to handle the fact that, in k-NN queries among parties sharing vertically partitioned data, it is likely the attributes may belong to very diverse domains. Each party may be applying a local metric M_{P_i} to the projection each holds of the two records $\vec{p}$ and $\vec{q}$. This results in the value $v_i = M_{P_i}(\vec{p},\vec{q})$. Thus, the global metric could be a weighted sum $\sum_{i=1}^m \omega_i v_i$ of the local metric values v_i. The problem of computing it would be solved by

[3] Note that, the <Alice vs Bob> comparison is not the same as the <Bob vs Alice> comparison. For instance say Alice's value is smaller than Bob's, then <Alice vs Bob> will output shares that add up to zero, but <Bob vs Alice> will output shares that should add up to one.

[4] Note that Bob does not need to compare with Alice again, he rather uses the second share provided from the third party.

[5] If all distances are different we know the maximum value is always $m - 1$. Our protocol works even if some comparisons are between equal values.

secure sum protocol. Alternatively, the global metric could be a weighted maximum $\max_{i=1}^{m} \omega_i v_i$. In this case, our Protocol 2 generalizes to compute the global metric from the local metric values on the attributes known to each corresponding party. This allows for very flexible metrics that take into account issues like different units of measure and data types on the attributes. One realizes that computing the Hamming distance $H(\vec{p}, \vec{q})$, which is the number of entries where the vectors $\vec{p}$ and $\vec{q}$ differ, reduces to computing again the sum of the Hamming distances in the projection by each party. If we now consider metrics, like the usage/access metrics or frequency metrics, we see that these metrics have the form $\vec{p}^T \cdot \vec{q}/\|\vec{p}\|_2\|\vec{q}\|_2$. That is, they are the cosine of the angle between the vectors $\vec{p}$ and $\vec{q}$. Here we are more interested in computing $\cos(\alpha) = \vec{p}^T \cdot \vec{q}/(\|\vec{p}\|_2\|\vec{q}\|_2)$ and split output in shares $s_a + s_b = \cos(\alpha)$ where again, s_a is know by Alice only and s_b is know by Bob only. Since dot products on vertically partitioned data are sums and can be computed with shares by secure sum [25], only Alice knows constants A_1, A_2 and A_3 and Bob knows only B_1, B_2 and B_3 so that

$$\begin{aligned}\cos(\alpha) &= \frac{A_1+B_1}{(A_2+B_2)(A_3+B_3)} = \frac{A_1+B_1}{A_2A_3+A_2B_3+B_2A_3+B_2B_3} \\ &= \frac{A_1+B_1}{A_2A_3+(A_2,A_3)^T\cdot\begin{pmatrix} B_3 \\ B_2 \end{pmatrix}+B_2B_3}\end{aligned}$$

Using the scalar product with shares [9], we obtain values A_4 and B_4 so that $A_4 + B_4 = (A_2, A_3)^T \cdot \begin{pmatrix} B_3 \\ B_2 \end{pmatrix}$, A_4 is known only to Alice and B_4 only to Bob. With $B_5 = B_4 + B_2B_3$ and $A_5 = A_2A_3 + A_4$ we have the derivation.

$$\cos(\alpha) = \frac{A_1 + B_1}{\underline{A_2A_3} + \underline{A_4} + \underline{B_4} + \underline{B_2B_3}} = \frac{A_1 + B_1}{A_5 + B_5}$$

where A_5 is only known by Alice and B_5 is known by Bob. For division that provides output with shares we propose the following protocol.

Protocol 3 *Alice holds (a_1, a_2) and Bob holds (b_1, b_2), Alice gets A and Bob B, with $A + B = (a_1 + b_1)/(a_2 + b_2)$.*

1. *Alice produces a random number r_1 and Bob produces a random number r_2.*

2. *Alice obtains $r_2(a_2+b_2) = (a_2, 1)^T \cdot \begin{pmatrix} r_2 \\ r_2b_2 \end{pmatrix}$ by using the scalar product protocol [9]* [6] *but provide an answer to one party only, and Bob obtains $r_1(a_2 + b_2) = (b_2, 1)^T \cdot \begin{pmatrix} r_1 \\ r_1a_2 \end{pmatrix}$.*

3. *Alice and Bob use again the scalar product [9], but provide an answer with shares A and B using the above input vectors from each party.*

$$A + B = \left(r_1a_1, \frac{1}{r_2(a_2+b_2)}\right)^T \cdot \begin{pmatrix} \frac{1}{r_1(a_2+b_2)} \\ r_2b_2 \end{pmatrix}$$

3.1 Private k-Nearest Neighbours

We now describe our first protocol for k-NN queries. Given vectors $\vec{v}_1 = (v_{11}, \cdots, v_{1m}), \cdots, \vec{v}_n = (v_{n1}, v_{n2}, \cdots, v_{nm})$, together with a vector $\vec{q} = (q_1, \cdots, q_m)$ (where m is the number of parties), we find $P(\vec{q}, k)$, where $P(\vec{q}, k)$ is the ids for the k-NN to $\vec{q}$. When $dist(\vec{p}, \vec{q}) = d^a_{pq} + d^b_{pq}$ and $dist(\vec{p}, \vec{r}) = d^a_{pr} + d^b_{pr}$ (with d^a_{pq}, d^a_{pr} known only to Alice and d^b_{pq}, d^b_{pr} known only to Bob), then $dist(\vec{p}, \vec{q}) < dist(\vec{p}, \vec{r})$ if and only if $\underline{d^a_{pq} - d^a_{pr} < d^b_{pq} - d^b_{pr}.}$

[6] Here Bob sets his private share $V_2 = 0$. As authors of this protocol remarked in the original paper [9](page 6), this does not let Alice to learn any of Bob's private data.

Protocol 4 *PP k-NN.*

1. *The parties calculate metrics with shares. After this, we can assume the first party Alice holds a vector $\vec{s}^a$ of dimension n and Bob holds a vector $\vec{s}^b$ so that $s^a_i + s^b_i = dis(\vec{q}, \vec{v}_i)$ (in fact, it is possible to assume encryption modulo a field F; i.e. $s^a_i + s^b_i = dis(\vec{q}, \vec{v}_i) \mod F$).*

2. *Alice computes the matrix D^A whose ij entry is $|s^a_i - s^a_j|$, while Bob computes the matrix $(D^B_{ij}) = (|s^b_i - s^b_j|)$.*

3. *Alice and Bob engage in Yao comparisons with share for each respective entry of D^A and D^B. Let σ^a_{ij} be Alice's share of comparing D^A_{ij} with D^B_{ij} (Bob's share is σ^b_{ij}).*

4. *Let Alice compute the vector V^A whose i-th entry is $\sum_{j=1}^{n} \sigma^a_{ij}$ while Bob's V^B is such that $V^B_i = \sum_{j=1}^{n} \sigma^b_{ij}$.*

5. *Alice and Bob use the secure add vectors protocol where Alice obtains $\pi^{-1}(V^A + V^B)$ with π known only to Bob.*

6. *Alice sorts and sends the top k fake-ids to Bob. Bob broadcast $\pi^{-1}(fakeIDs)$ (IDs for k-NN of $\vec{q}$).*

Note that, Alice only learns counts of $d^a_{pq} - d^a_{pr} < d^b_{pq} - d^b_{pr}$ comparisons. The security proof follows from the composition theorem. This protocol is quadratic on n. The following protocol is $O(n \log n)$.

Protocol 5 *Fast PP k-NN.*

1. *Same as Protocol 4.*

2. *Bob (the second party) uses a random value R_b only know to him and adds the vector $\vec{R}_b = (R_b, ..., R_b)$ (that consists of all entries set to one random number R_b). He adds $\vec{s}^b + \vec{R}_b$.*

3. *Alice and Bob use the add-vectors protocol for $\vec{s}^a$ known to Alice and $\vec{s}^b + \vec{R}_b$ known to Bob. This gives Alice a random translation of the distances $dist(\vec{q}, \vec{v}_i)$, for $i = 1, ..n$, permuted by a random permutation π that only Bob knows.*

4. *Alice sorts and sends the top k fake-ids to Bob. Bob broadcast $\pi^{-1}(fakeIDs)$ (IDs for k-NN of $\vec{q}$).*

We remark that Alice learns the distribution of the distance values translated by a random number, which we believe is innocuous information. No other information is disclosed. Also if we use the versions with shares over a field for computing metrics, then it is possible to use binary search over a field for the top k-neighbours [24]. The field binary search requires Yao's comparisons with shares. This would protect from Alice learning the distribution of the distance values at the expense of as many as $O(n \log |F|)$ Yao comparisons.

4 The Private $SASH$ Data Structure

While above algorithms for the k-NN query adapts them in the privacy-preserving context, these algorithms should be used for small data sets, since its complexity is proportional to the number n of data vectors. For large data sets, we should use data structures that allows much better performance on the number N of items in the database. This can be illustrated by the role that R-Trees play in the efficiency $DBSCAN$ [1]. To manage very large data sets, we provide a privacy-preserving $SASH$. The $SASH$ [16, 15] makes minimal assumptions about the nature of the metric for associative queries. It also does not enforce a partition of the search space, as for example R-Trees do. A partition of the space enables the parties to learn bounding boxes for other' data [2] — a less private

Average Size S of the union for $A0$ algorithm number m of parties					$SASH$
k	4	6	8	10	
10	2,982±218	3,694±138	4,535±136	5,092±117	952
20	3,100±203	4,037±141	4,835±106	5,438±55	952
25	3,429±161	4,270±127	5,004±98	5,448±69	952
50	3,704±150	4,689±125	5,294±68	5,628±44	989

Table 2. Performance of A0 and $SASH$ on dataset "The Insurance Company Benchmark (CoIL 2000)".

approach. For approximate k-NN (k-ANN) queries on the large sets, the $SASH$ consistently returns a high proportion of the true k-NNs at speeds of roughly two orders of magnitude faster than sequential search [15]. The $SASH$ has already been successfully applied to clustering and navigation of very large, very high dimensional text data sets [14]. However, the current form of the $SASH$ would be inappropriate for privacy preservation. The $SASH$ internally uses a k-NN query on small sets. We replace this internal k-NN query, by our algorithm for the privacy preserving context. This approach has been used before [2]. The end result is a data structure for the privacy-preserving context.

5 Performance Evaluation

We show that for our protocols and algorithms, the cost is essentially as for a distributed non-private setting ($DNPS$) where parties would still need to incur local calculations and communication costs between them or to a central party. Also, the implementation of our privacy-preserving algorithms is feasible. Using our Yao-comparison with shares and our Protocol 2 increases the complexity of the Chessboard distance to quadratic in the number of parties. However, the number of parties would be of the order of about 10. Thus, very affordable for the additional privacy. For the $Minkowski$ metrics, the main cost is the communication cost, which is comparable to $DNPS$. In fact any other local cost (computing local projections of the metric) would also be performed in $DNPS$. For the combination metrics (like sum/max of local metrics) the performance will be the same as for the $Minkowski$ metrics for sums and Chessboard for maximums. Note that, like in the overwhelming majority of methods for k-NN queries and associative queries, the major cost is not the computation of distances per se, but how many of these computations are performed. The only possible competitor to our privacy-preserving $SASH$ method to perform k-NN queries is the privacy-preserving version [24] of Fagin's A0 algorithm [10]. However, for a vertically partitioned database with N records, this algorithm's complexity (time and communication cost) includes as a factor the number S of candidates generated. It is well recognised that S can be as large as N and in the best case as small as k. The accepted [10, 24] worst-case theoretical analysis is that the complexity is $O(N^{(m-1)/m}k^{1/m})$ where m is the number of parties. However, there are no studies on what is the expected performance of this algorithm. We evaluated the size S of the union of Fagin's AO algorithm in 5 well-known large data sets. The CoIL 2000 Challenge [26] contains data that consists of 86 variables. We repeated the following experiment 100 times. We partition the attributes randomly into m parties, we selected random metrics for each party (among Euclidean, Hamming, Chessboard and $Minkowski$ with $r = 1$), we selected a random query point from the data and computed the k-NN using Fagin's A0 algorithm. The Table. 2 shows the average size S of the union in Fagin's AO algorithm for this data set with 95% confidence intervals. This data set has $5,822$ records and we can see that most of the entries of the table are above $3,000$ while several are above the $5,000$. It is disappointing that when asking for 10 neighbours among 8 parties we expect a union size to be 78% of the size N of the database. Note that the worst case for all query sizes k is above $5,000$ and that the size of the union in the best observed case is well above $500 \times k$, and rapidly above 50% the size of the file. Similar results occur for the Census-Income Database holding multivariate PUMS census data (from KDD UCI repository). The inefficiency of AO is also reflected in three large datasets previously used for k-NN queries [11]. The data set named *Histogram* corresponds to a color histogram, while the one named *Stock* corresponds to a stock market price. Finally, an aerial image dataset was tested, which gave similar results. Another reason for performing this analysis is that the privacy-preserving version of Fagin's AO algorithm [24] leaks all the ids of the union. Therefore, the size S of the union not only determines the inefficiency of the method but is also a strong measure of the lack of security in the algorithm. The privacy-preserving AO algorithm will perform at least as many distance evaluations as the number S of candidates (or the size of the union). We have chosen the $SASH$ because this data structure is very efficient invoking a number of distance computations which is bounded by $pcN \log 2N$ [16, 15] (for construction), while the bound for an approximate k-NN query under the uniform search is $ck \log 2N$, and $\frac{k^{1+\frac{1}{\log 2N}}}{k^{\frac{1}{\log 2N}}-1} + 2p^3 \log 2N$ for geometric search (here p and c are the constant parameters of the $SASH$ and k is the number of NN requested). Therefore, the $SASH$ will easily outperform Fagin's A0 algorithm, and will provide logarithmic response for k-NN queries and associative queries. The privacy-preserving $SASH$ invokes the Protocol 5 for small sets of size n. Our protocol does require $\Theta(n \log n)$ time on Alice side for sorting and $O(n)$ time on Bob side to add a random value to all the shares it holds and to generate the random permutation π. However, the constants involved are small and clearly the process is practical. However, the search in a field requires $(n+1)\log|F|$ rounds [24]. Typical values are $F = 10^6$, which makes a larger constant in the $O(n)$ for this alternative. Clearly, the local computation is far more in this aspect alone that our algorithm. In terms of communication cost, our approach is also more efficient. Bob sends exactly n values, and Alice sends back k values. The binary search in a field performs the communications needed for $(n+1)\log|F|$ Yao-comparisons. Overall, Protocol 5's complexity depends on the number m of parties, the number n of vectors and the number k of near neighbors we are looking for. The implementation confirms that the performance cost is dominated by $\Theta(n \log n)$ time. For communication cost, the $n + k$ complexity is dominated by n again.

We have also implemented the $SASH$ method and evaluated the performance on the COIL and PUMS data sets. The performance depends directly on the number of candidates generated at all levels of the $SASH$(the sum of $\|P_i(q, k_i)\|$ for all levels [15, page 8]). Our results for COIL are the last column in Table 2. The $SASH$ candidate generation does not depend on m, and shows remarkable small numbers for both data sets (about 12% for COIL and 0.5% for PUMS). This demonstrates the efficiency of the $SASH$ approach.

References

[1] A. Amirbekyan and V. Estivill-Castro. Privacy preserving $DBSCAN$ for vertically partitioned data. In *IEEE ISI*, p. 141–153, San Diego, CA, USA, 2006.

[2] A. Amirbekyan and V. Estivill-Castro. The privacy of k-nn retrieval for horizontal partitioned data — new methods and applications. *(ADC2007)*, v. 63, p. 33–42, Ballarat, Victoria, Australia, 2007.

[3] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. In *5th ACM-SIAM Symposium on Discrete Algorithms*, p. 573–582, 1994.

[4] S. Avidan and M. Butman. Blind vision. *Computer Vision - ECCV 2006, Part III*, volume 3953 of *LNCS*, p. 1–13, Graz, Austria, 2006.

[5] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful ? In *ICDT*, p. 217–225, Jerusalem, Israel, 1999.

[6] C. Cachin. Efficient private bidding and auctions with an oblivious third party. In *Proc. of the 6th ACM Conference on Computer and Communications Security*, p. 120–127, Singapore, 1999.

[7] P. Ciaccia and M. Patella. PAC nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. In *Proc. of the ICDE*, p. 244–255, San Jose, CA, 2000.

[8] W. Du and M.J. Atallah. Privacy-preserving cooperative statistical analysis. In *Proc.of the (ACSAC)*, p. 102–110, New Orleans, Louisiana, 2001.

[9] W. Du and Z. Zhan. Building decision tree classifier on private data. *Privacy, Security and Data Mining*, p. 1–8, Sydney, Australia, 2002. IEEE ICDM Workshop Proceedings.

[10] R Fagin. Combining fuzzy information from multiple systems. In *Proceedings of the ACM Symposium on Principles of Database Systems*, p. 216–226, Montreal, Canada, 1996.

[11] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. El Abbadi. Approximate nearest neighbor searching in multimedia databases. In *7th IEEE (ICDE)*, p. 503–511, Germany, 2002.

[12] O. Goldreich. *The Foundations of Cryptography*, volume 2, chapter General Cryptographic Protocols. Cambridge University Press, 2004.

[13] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game (extended abstract). *Proc. of the 19th ACM Annual Symposium on Theory of Computing*, p. 218–229, New York, 1987.

[14] M.E. Houle. Navigating massive data sets via local clustering. *9th ACM SIGKDD*, p. 547–552, Washington, DC, 2003.

[15] M.E. Houle. SASH: a spatial approximation sample hierarchy for similarity. Technical Report RT-0517, IBM Tokyo Research Laboratory, 2003.

[16] M.E. Houle and J. Sakuma. Fast approximate similarity search in extremely high-dimensional data sets. In *ICDE*, p. 619–630, Tokyo, Japan, 2005.

[17] I. Ioannidis and A. Grama. An efficient protocol for Yao's millionaires' problem. In *36th HICSS*, p. 205, Big Island, HI, 2003.

[18] M. Kantarcioğlu and C. Clifton. Privately computing a distributed k-nn classifier. *PKDD*, v. 3202, p. 279–290. LNCS, 2004.

[19] J. Mena. *Investigative Data Mining for Security and Criminal Detection*. Butterworth-Heinemann, US, 2003.

[20] N. Roussopoulos, S. Kelly, and F. Vincent. Nearest neighbor queries. In *Proceedings of the ACM SIGMOD*, p. 71–79, San Jose, CA, 1995.

[21] M. Shaneck, Y. Kim, and V. Kumar. Privacy preserving nearest neighbor search. 2006 IEEE ICDM Workshop on Privacy Aspects of Data Mining, 2006.

[22] J. Vaidya and C. C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the SIGKDD-ACM*, p. 206–215, Washington, DC, 2003.

[23] J. Vaidya and C. Clifton. Privacy-preserving outlier detection. In *4th IEEE (ICDM)*, Brighton, UK, 2004.

[24] J. Vaidya and C. Clifton. Privacy-preserving top-k queries. In *ICDE* , p. 545–546, Tokyo, 2005.

[25] J. Vaidya, C. Clifton, and M. Zhu. *Privacy Preserving Data Mining*, volume 19 of *Advances in Information Security*. Springer, New York, 2006.

[26] P. van der Putten and M. van Someren. CoIL challenge 2000: The insurance company case. Technical Report 2000-09, Leiden Institute of Advanced Computer Science, Amsterdam, 2000.

[27] A.C. Yao. Protocols for secure computation. In *IEEE Symposium of Foundations of Computer Science*, p. 160–164. 1982.